The MLIP package: moment tensor potentials with MPI and active learning

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 025002 https://doi.org/10.1088/2632-2153/abc9fe

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
6 August 2020

REVISED
29 October 2020

ACCEPTED FOR PUBLICATION
12 November 2020

PUBLISHED
28 December 2020

PAPER

The MLIP package: moment tensor potentials with MPI and active learning

Ivan S Novikov[1], **Konstantin Gubaev**[1,2], **Evgeny V Podryabinkin**[1] **and Alexander V Shapeev**[1]

[1] Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Nobel St. 3, Moscow 143026, Russian Federation
[2] Department of Materials Science and Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands

E-mail: a.shapeev@skoltech.ru

Keywords: machine-learning interatomic potentials, active learning, *ab initio* calculations

Abstract

The subject of this paper is the technology (the 'how') of constructing machine-learning interatomic potentials, rather than science (the 'what' and 'why') of atomistic simulations using machine-learning potentials. Namely, we illustrate how to construct moment tensor potentials using active learning as implemented in the MLIP package, focusing on the efficient ways to automatically sample configurations for the training set, how expanding the training set changes the error of predictions, how to set up *ab initio* calculations in a cost-effective manner, etc. The MLIP package (short for Machine-Learning Interatomic Potentials) is available at https://mlip.skoltech.ru/download/.

1. Introduction

Machine-learning interatomic potentials have recently been a subject of research and now they are turning into a *tool* of research. Interatomic potentials (or force-fields) are models predicting potential energy (together with its derivatives) of interaction of a variable-size atomic system as a function of atomic positions and types. Machine-learning potentials differ from classical, empirical potentials such as EAM or REAX-FF by a flexible functional form that can be systematically improved to approximate an arbitrary quantum-mechanical interaction subject to some assumptions like locality of interaction or smoothness of the underlying potential energy surface. For about a decade, the focus of research has been on feasibility and properties of such an approximation—its accuracy, efficiency, and transferability depending on the chemical composition, nature of bonding, or level of quantum-mechanical theory being approximated. In contrast, this manuscript describes how to practically apply the moment tensor potentials (MTPs) to seamlessly accelerate quantum-mechanical calculations, and the MLIP package (short for Machine-Learning Interatomic Potentials) implementing this tool.

The concept of machine-learning potentials was pioneered in 2007 by Behler and Parrinello [1], who, motivated by the success of approximating potential energy surfaces of small molecules with neural networks, proposed the use of neural networks as a functional form of interatomic potentials, going beyond the small molecular systems by employing the locality of interaction. The structure of such potentials consists of two parts: descriptors—usually two- and three-body ones—whose role is to describe local atomic environments accounting for all the physical symmetries of interaction, and a regressor—a function that maps the descriptors onto the interaction energy. Neural network has been the first and are probably the most popular form of regressor for machine-learning potentials [2–8]. A related but slightly different class of potentials are the message-passing potentials that are based on neural networks, but go beyond the local descriptors of atomic environments [9, 10].

The other form of regressor proposed in 2010 was Gaussian processes used in the Gaussian approximation potentials (GAP) [11–15]. When used with the smooth overlap of atomic positions (SOAP) kernel [15, 16], they can provably approximate an arbitrary local many-body interaction of atoms, in

contrast to the neural network potentials employed on top of two- and three-body descriptors [17][3]. An alternative application of Gaussian processes was recently proposed [18] for explicit three-body potentials.

Probably the third largest class of interatomic potentials is based on linear regression with a set of basis functions. This includes the spectral neighbor analysis potential (SNAP) [19, 20] including the recent extension to multicomponent systems [21], and polynomial-based approaches [22, 23] including MTP [24]—the main focus of the present work. Our extension of MTP to multicomponent systems [25, 26] goes beyond the linear regression, however, there is an alternative formulation of multicomponent MTP that stays linear [27].

Standing aside from these three classes are the interatomic potentials based on symbolic regression [28]. In contrast to the trend of choosing a functional form with increasing the number of parameters to reach a certain approximation accuracy, the symbol-regression potentials assume a very general functional form (essentially, limiting itself to a mathematical formula with a certain set of operations) and try to reach the desired accuracy with as few parameters as possible. The advantage of such potentials is improved transferability which in the case of MTPs we ensure by active learning as will be outlined below.

There exist related approaches solving similar problems but falling outside the class of machine-learning potentials. These are non-conservative force fields [29, 30], force-fields for a fixed-size molecular system (typically applied to one or two organic molecules) [31, 32], on-lattice potentials [33, 34], and cheminformatics-type models [35–43]; see also a recent review [44].

1.1. Moment tensor potentials

MTP has been first proposed as a single-component potential with linear dependence of the energy on the fitting parameters [24]. The basis in which the potential was expanded is polynomial-like (adapted so that instead of exhibiting polynomial growth at large interatomic distances the basis functions stop feeling atoms that left the finite cutoff sphere). This basis is very similar to the one of atomic cluster expansion (ACE), [22] and related to the permutation-invariant polynomial (PIP) basis [23]. An active learning algorithm was proposed in [45] based on which point defect diffusion [46] and lattice dynamics [47] of a number of mono-component crystals were studied. Later MTP was generalized to multiple components by introducing non-linear dependence on some of the parameters—first in the context of cheminformatics [26] and later as an interatomic potential [25]. Moment tensor potential was applied to predicting stable crystalline structures of single-components [48] and convex hulls of stable alloy structures [25], simulating lattice dynamics [47], calculating free energy of high-entropy alloy [49], calculating mechanical properties of a medium-entropy alloy [50], computing lattice thermal conductivity of complex materials [51], studying diffusion of point defects in crystals [46] and Li atoms in cathode coating materials [52], modeling covalent and ionic bonding [53], computing molecular reaction rates [54, 55], and studying a variety of properties of two-dimensional materials [56–59], including enabling multiscale calculations of heat conductivity of polycrystalline materials [60] which are otherwise hard to carry out with classical, pre-Big Data modeling approaches.

MTPs, like SOAP-GAP, are not based on solely two- and three-body descriptors and can provably approximate an arbitrary local interaction, and so are ACE [22] and PIP [23]. Probably because of this MTP together with GAP showed excellent accuracy in recent cheminformatics benchmark test [61] and interatomic potential test [62]; in the latter MTP showed also a very good balance between accuracy and computational efficiency when compared against other machine-learning potentials.

1.2. Active learning

A crucial and often time-consuming part is the construction of the training set. Traditionally, the training set is constructed through laborious trial-and-error iterations in each of which a researcher manually assesses the performance of the trained potential and tries to understand how to construct configurations for the new training set to avoid the undesirable behavior of the potential on the next iteration. Active learning is a machine-learning technique allowing one to entrust these training set refinement iterations to a computer, thus completely automating the training set construction.

The ideas of active learning during atomistic simulations come from the concept of learning on-the-fly [63, 64] which can be retrospectively described as a learning-and-forgetting scheme. In this scheme the training set would consist of some of the last configurations of the molecular dynamics trajectory. A learning-and-remembering scheme was first proposed in [30], however, the sampling was uniform—a fixed number of time steps were skipped before adding a configuration into the training set, which otherwise grows indefinitely.

[3] Strictly speaking, [17] proves that *not every* partitioning of the total energy into local contributions can be approximated as a function of two- and three-body descriptors. The extent to which this affects the accuracy of approximating the *best* partitioning is yet to be investigated.

For a more sophisticated strategy one needs an indicator of an error that a machine-learning model commits when trying to predict the energy and derivatives of a configuration *without* making a quantum-mechanical calculation. Such an indicator in the field of machine learning is called a *query strategy* [65]. Such a strategy was first implemented by Artrith and Behler in [2] for the neural network potential. In their work they used an ensemble of independently trained neural networks to 'vote' for the predicted energy. The deviation between different neural networks was taken as the sought indicator of the error. It was then used to conduct a molecular dynamics simulation and adding to the training set those configurations on which different neural networks significantly disagreed. This query strategy is called the *query by committee*. It is the algorithm of choice for other neural-network-based potentials as well [66, 67]. An exception is the work [68] where the authors train two models: one predicting the energy and another predicting uncertainty.

Gaussian process-based potentials have another natural query strategy—predictive variance [14, 18]. Interestingly, for the original GAP other sampling criteria has been proposed [69, 70].

We, for the MTP, employ a special form of what is known as the D-optimality criterion [65]. It employs a geometric criterion based on the so-called *extrapolation grade*—a quantity characterizing the extent to which a given configuration is extrapolative with respect to those in the training set. This algorithm was proposed for linearly parametrized, single-component MTPs in [45] and generalized to non-linear MTPs in [25, 26]. The details of our algorithm will be given in section 2.3.

1.3. Structure of this manuscript

In this manuscript we focus on the methodology of applying MTPs and active learning to performing atomistic simulations. In section 2 we describe our formulation of the moment tensors potentials and active learning. Section 3 gives a brief overview of the MLIP package implementing MTPs and active learning, which is detailed in [71]. Then, sections 4–6 describe three practical examples of the use of MTPs and active learning to perform atomistic calculations, with the focus on the particular steps of the calculation. The files that are referenced in sections 4–6 are available in our supplementary material [72], as well as in the `doc/examples/` folder of the MLIP package [73]. The description of and references to the MLIP code is kept to minimum, yet retained for the purpose of easier preproduction of the described results.

2. Theory

2.1. Moment tensor potential

MTP belong to the class of machine-learning potentials implemented in the MLIP package. These potentials represent the energy of an atomic configuration `cfg` as a sum of contributions of local atomic environments of each atom. The atomic environment, or neighborhood, $\mathfrak{n}_i$ of the ith atom is comprised of its atomic type z_i, the atomic type of its neighbors, z_j and positions of the neighbors relative to the ith atom, $\mathbf{r}_{ij}$. The potential energy of interatomic interaction, E^{mtp}, is thus

$$E^{\mathrm{mtp}}(\mathrm{cfg}) = \sum_{i=1}^{n} V(\mathfrak{n}_i). \tag{1}$$

The function V is linearly expanded through a set of basis functions B_α:

$$V(\mathfrak{n}_i) = \sum_{\alpha} \xi_\alpha B_\alpha(\mathfrak{n}_i), \tag{2}$$

where $\xi = \{\xi_\alpha\}$ are parameters to be found by fitting them to the training set.

To define the functional form of the basis functions B_α, we introduce the moment tensor descriptors, or simply *moments*:

$$M_{\mu,\nu}(\mathfrak{n}_i) = \sum_{j} f_\mu(|r_{ij}|, z_i, z_j) \underbrace{\mathbf{r}_{ij} \otimes \ldots \otimes \mathbf{r}_{ij}}_{\nu \text{ times}};$$

the notation is detailed below. These descriptors consist of the radial and angular part. The radial part has the form

$$f_\mu(|r_{ij}|, z_i, z_j) = \sum_{\beta=1}^{N_Q} c^{(\beta)}_{\mu,z_i,z_j} Q^{(\beta)}(|r_{ij}|), \tag{3}$$

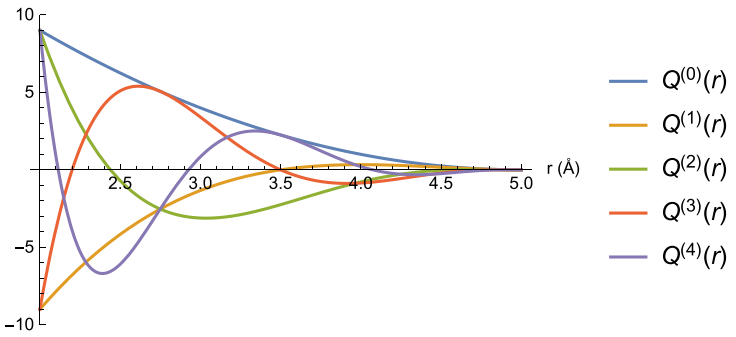


Figure 1. Radial basis functions $Q^{(\beta)}(r)$ as defined in (4). Plotted for $R_{\min} = 2$ Å, $R_{\text{cut}} = 5$ Å, $0 \le \beta \le 4$.

where $\mathbf{c} = \{c^{(\beta)}_{\mu,z_i,z_j}\}$ is the set of 'radial' parameters. We call the functions $Q^{(\beta)}(|r_{ij}|)$ the radial basis functions:

$$Q^{(\beta)}(|r_{ij}|) = \begin{cases} \varphi^{(\beta)}(|r_{ij}|)(R_{\text{cut}} - |r_{ij}|)^2 & |r_{ij}| < R_{\text{cut}} \\ 0 & |r_{ij}| \ge R_{\text{cut}}. \end{cases} \tag{4}$$

Here $\varphi^{(\beta)}$ are polynomial functions (e.g. Chebyshev polynomials) on the interval $[R_{\min}, R_{\text{cut}}]$, where $R_{\min}$ is the minimal distance between atoms in the system investigated, R_{cut} is the cutoff radius which is introduced to ensure a smooth behavior of MTP when atoms leave or enter the interaction neighborhood. An illustration of the radial basis functions is given in figure 1.

The angular part $\underbrace{\mathbf{r}_{ij} \otimes \ldots \otimes \mathbf{r}_{ij}}_{\nu \text{times}}$ contains angular information about the neighborhood $\mathbf{n}_i$. Here the symbol '$\otimes$' is the outer product of vectors, and, thus, the angular part is the tensor of rank ν. E.g. for $\nu = 0$ the angular part is a scalar and simply equals 1, if $\nu = 1$ the angular part is the vector $\mathbf{r}_{ij} = (x_{ij}, y_{ij}, z_{ij})$ pointing from atom i to atom j, if $\nu = 2$ the angular part has the form of the matrix:

$$\mathbf{r}_{ij} \otimes \mathbf{r}_{ij} = \begin{pmatrix} x_{ij}^2 & x_{ij}y_{ij} & x_{ij}z_{ij} \\ y_{ij}x_{ij} & y_{ij}^2 & y_{ij}z_{ij} \\ z_{ij}x_{ij} & z_{ij}y_{ij} & z_{ij}^2 \end{pmatrix}.$$

In order to construct the basis functions B_α we define the so-called *level* of moments:

$$\text{lev}M_{\mu,\nu} = 2 + 4\mu + \nu, \tag{5}$$

for example $\text{lev}M_{0,1} = 3$, $\text{lev}M_{1,1} = 7$, $\text{lev}M_{0,2} = 4$, $\text{lev}M_{0,0} = 2$. The coefficients 2, 4, and 1 in (5) were empirically found to be optimal on a number of tests done in [25] and are fixed in the MLIP package. The level of multiplication, or more generally, contractions of a number of moments is defined by adding the levels, for example

$$\begin{aligned} \text{lev}M_{1,0}^2 &= 12, & \text{lev}M_{0,0}^4 &= 8, & \text{lev}M_{2,0}^3 &= 30, \\ \text{lev}(M_{1,1} \cdot M_{0,1}) &= 10, & \text{lev}(M_{1,2} : M_{0,2}) &= 12, & \text{lev}((M_{0,3}M_{0,2}) \cdot M_{0,1}) &= 12, \end{aligned}$$

where '$\cdot$' is the dot product of two vectors, ':' is the Frobenius product of two matrices.

As could be seen from these examples, non-scalar moments could yield scalars upon their contraction. All such contractions of one or more moments are, by definition, MTP basis functions B_α. Each basis function is, by its definition, invariant to atomic permutations, rotations, and reflections. Finally, to define a particular functional form of MTP, we choose the maximum level, $\text{lev}_{\max}$, and include all the basis functions whose level is less or equal than that: $\text{lev}B_\alpha \le \text{lev}_{\max}$. For example, if we choose $\text{lev}_{\max} = 8$ then we have nine basis functions B_α:

$$\begin{aligned}
B_1 &= M_{0,0}, & \mathrm{lev}B_1 &= 2;\\
B_2 &= M_{1,0}, & \mathrm{lev}B_2 &= 6;\\
B_3 &= M_{0,0}^2, & \mathrm{lev}B_3 &= 4;\\
B_4 &= M_{0,1}\cdot M_{0,1}, & \mathrm{lev}B_4 &= 6;\\
B_5 &= M_{0,2} : M_{0,2}, & \mathrm{lev}B_5 &= 8;\\
B_6 &= M_{0,0}M_{1,0}, & \mathrm{lev}B_6 &= 8;\\
B_7 &= M_{0,0}^3, & \mathrm{lev}B_7 &= 6;\\
B_8 &= M_{0,0}(M_{0,1}\cdot M_{0,1}), & \mathrm{lev}B_8 &= 8;\\
B_9 &= M_{0,0}^4, & \mathrm{lev}B_9 &= 8.
\end{aligned} \tag{6}$$

The radial parameters **c** from (3) together with ξ from (2) comprise the set of MTP parameters $\theta = \{\xi, \mathbf{c}\}$ that are found by fitting to the training set as described in the next section. Thus, the energy as predicted by MTP will be denoted as $E^{\mathrm{mtp}}(\mathrm{cfg};\theta)$ when we want to emphasize the dependence on θ.

Thus, the functional form of MTP is determined by the two numbers—level of MTP $\mathrm{lev}_{\max}$ and size of the radial basis N_Q defined in (3). The number of basis functions (and the number of the corresponding parameters ξ) grows exponentially with $\mathrm{lev}_{\max}$, while the number of radial functions f_μ (and the number of the corresponding parameters **c**) grows as $O(N_Q \mathrm{lev}_{\max})$. The hyperparameters $\mathrm{lev}_{\max}$ and N_Q should be chosen, for a particular application, to achieve the desired balance between the accuracy of MTP, computational efficiency of MTP, and number of the required quantum-mechanical calculations, the latter is usually proportional to the total number of free parameters in MTP.

2.2. Training on a quantum-mechanical database

Let the training set contain the configurations cfg_k, $k = 1, \ldots, K$, with known quantum-mechanical energies $E^{\mathrm{qm}}(\mathrm{cfg}_k)$, forces $\mathbf{f}_i^{\mathrm{qm}}(\mathrm{cfg}_k)$, and stress tensors $\sigma^{\mathrm{qm}}(\mathrm{cfg}_k)$. The passive learning (fitting, training) of MTP consists of finding the parameters θ during solving the machine-learning (optimization) problem:

$$\begin{aligned}
\sum_{k=1}^{K}\Big[w_e\left(E^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta) - E^{\mathrm{qm}}(\mathrm{cfg}_k)\right)^2 + w_f\textstyle\sum_{i=1}^{N_k}\left|\mathbf{f}_i^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta) - \mathbf{f}_i^{\mathrm{qm}}(\mathrm{cfg}_k)\right|^2\\
+w_s\left|\sigma^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta) - \sigma^{\mathrm{qm}}(\mathrm{cfg}_k)\right|^2\Big] \to \min_\theta,
\end{aligned} \tag{7}$$

where N_k is the number of atoms in the kth configuration, w_e, w_f, and w_s are non-negative weights expressing the importance of energies, forces, and stresses in the optimization problem. Here for a stress tensor σ by $|\sigma|^2$ we mean the Frobenius norm, $|\sigma|^2 = \sum_{\alpha,\beta=1}^{3}|\sigma_{\alpha\beta}|^2$.

After training we can measure the root-mean-square errors in energy, forces, and stresses as

$$\mathrm{RMSE}(E)^2 = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{E^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta)}{N^{(k)}} - \frac{E^{\mathrm{qm}}(\mathrm{cfg}_k)}{N^{(k)}}\right)^2, \tag{8}$$

$$\mathrm{RMSE}(\mathbf{f})^2 = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{3\,\mathrm{N}^{(k)}}\sum_{i=1}^{N_k}\left|\mathbf{f}_i^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta) - \mathbf{f}_i^{\mathrm{qm}}(\mathrm{cfg}_k)\right|^2, \tag{9}$$

$$\mathrm{RMSE}(\sigma)^2 = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{9}\left|\sigma^{\mathrm{mtp}}(\mathrm{cfg}_k;\theta) - \sigma^{\mathrm{qm}}(\mathrm{cfg}_k)\right|^2. \tag{10}$$

The mean absolute errors and maximum absolute errors are defined accordingly. It is often a good idea to set aside a validation (or test) set of configurations on which to measure the errors according to the formulae (8)–(10)—this would give an unbiased, unaffected by overfitting, estimate of the error of the potential.

Depending on the variety and size of configurations in the training set we may consider different weighting of configurations depending on the number of atoms, N_k, in the optimization problem. The main criterion is that the same configuration with a larger unit cell should have the same relative contributions of macroscopic properties (energy and stresses), and microscopic properties (forces). Thus, if the training set consists only of configurations of not more than tens of atoms and in which unit cell (and, also, stresses) does not play an important role (e.g. organic molecules), we may find MTP parameters by solving the problem 7 with $w_s = 0$, and without any scaling by N_k. We tag the weighting given by 7 as 'molecules'.

If we have configurations of different size in the training set (the reader may think of different supercells of the same structure) but we want all the structures to have the same weight in the training set, irrespective of the number of atoms that are used to represent it, then we use the following scaling (tagged as 'structures'):

$$\sum_{k=1}^{K}\left[\frac{w_\mathrm{e}}{(N_k)^2}\left(E^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-E^\mathrm{qm}(\mathrm{cfg}_k)\right)^2+\frac{w_\mathrm{f}}{N_k}\sum_{i=1}^{N_k}\left|\mathbf{f}_i^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-\mathbf{f}_i^\mathrm{qm}(\mathrm{cfg}_k)\right|^2\right.$$
$$\left.+\frac{w_\mathrm{s}}{(N_k)^2}\left|\sigma^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-\sigma^\mathrm{qm}(\mathrm{cfg}_k)\right|^2\right]\to\min. \tag{11}$$

Finally, if we are studying thermal properties with molecular dynamics then we assume that in two configurations with different number of atoms each force vector has the same importance and the importance of the energy grows as $\sqrt{N_k}$, hence in this case we consider the following scaling (tagged as 'vibrations'):

$$\sum_{k=1}^{K}\left[\frac{w_\mathrm{e}}{N_k}\left(E^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-E^\mathrm{qm}(\mathrm{cfg}_k)\right)^2+w_\mathrm{f}\sum_{i=1}^{N_k}\left|\mathbf{f}_i^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-\mathbf{f}_i^\mathrm{qm}(\mathrm{cfg}_k)\right|^2\right.$$
$$\left.+\frac{w_\mathrm{s}}{N_k}\left|\sigma^\mathrm{mtp}(\mathrm{cfg}_k;\theta)-\sigma^\mathrm{qm}(\mathrm{cfg}_k)\right|^2\right]\to\min. \tag{12}$$

We refer to the method described in this section as passive learning of MTP because here we generate our training set manually and MTP is not 'choosing' what to train itself on. Our MLIP code also allows one to select configurations for the training set automatically, using the so-called active learning algorithm, presented in the next section.

2.3. Active learning: query strategy

The quality of a machine-learning potential is determined not only by a functional form (an efficient representation) but also by the quality of the training set. A known drawback of machine-learning potentials is poor prediction for configurations that are far from the training set (see, e.g. [15]). This could be described by saying that we should assemble the training set such that during the simulation (e.g. molecular dynamics) the potential is 'interpolating' with respect to the training set when trying to make predictions for energy, forces, and stresses. An extrapolation may lead to significant errors causing instability of atomistic simulation. For example, it is hard to expect an accurate simulation of a free surface with a potential fitted only on the bulk configurations. The notion of interpolation and extrapolation can even be rationalized mathematically [45].

Traditionally, the training set for interatomic potentials is constructed manually through several loops of trial-and-error, manually assessing the quality of the potential. However, with a formal definition of extrapolation it is possible to automate this procedure reducing months of manual work to hours of computer time. Our definition of extrapolation is based on the D-optimality criterion [45] postulating that a good training set is the one that corresponds to the maximal value of the determinant of the information matrix [65]. To be precise, we introduce the notion of extrapolation grade $\gamma(\mathrm{cfg})$—a feature of a configuration (and a training set) that correlates with the prediction error but does not require *ab initio* information to be calculated prior to its evaluation; its precise definition is given below.

Once we have defined the extrapolation grade $\gamma(\mathrm{cfg})$, we can formulate our *query strategy*: we collect all the configurations occurring in a simulation whose grade γ is higher than a chosen threshold. Such configurations are later computed with an *ab initio* model and added to the training set. This dynamically ensures that during a simulation no significant extrapolation occurs.

The D-optimality criterion that MLIP is based on, is easiest to understand in the context of linear regression, i.e. in the case when MTP is parametrized only by linear parameters ($\xi=\{\xi_1\dots\xi_m\}$). According to (1) and (2) the energy of a configuration in this case can be expressed as

$$E^\mathrm{mtp}(\mathrm{cfg};\xi)=\sum_i\sum_{\alpha=1}^{m}\xi_\alpha B_\alpha(\mathfrak{n}_i)=\sum_{\alpha=1}^{m}\xi_\alpha\underbrace{\sum_i B_\alpha(\mathfrak{n}_i)}_{b_\alpha(\mathrm{cfg})}.$$

When fitting to the energy values, we need to solve an overdetermined system of K linear equations on ξ with the matrix

$$\mathsf{B}=\begin{pmatrix}b_1(\mathrm{cfg}_1) & \dots & b_m(\mathrm{cfg}_1)\\ \vdots & \ddots & \vdots\\ b_1(\mathrm{cfg}_K) & \dots & b_m(\mathrm{cfg}_K)\end{pmatrix}. \tag{13}$$

Each equation of this system is produced by a certain configuration. In our version of the D-optimality criterion we select m configurations that yield a set of the most linearly independent equations in the sense that the corresponding $m \times m$ submatrix $\mathtt{A}$ has the maximal modulus of determinant, $|\det(\mathtt{A})|$ (maximal volume). We call the m selected configurations the *active set* and in the MLIP code the active set together with $\mathtt{A}$ and $\mathtt{A}^{-1}$ is called the *active learning state*. Note that the D-optimal active set corresponds to the most 'extreme' and diverse configurations from the point of view of MTP.

A practical way to construct an active set from a pool of configuration is provided by the Maxvol algorithm for finding the submatrix of maximal volume in a tall matrix [74]. Assume that we already have some active set of configurations and the corresponding rows of the square matrix $\mathtt{A}$. We can then test whether a candidate configuration can increase $|\det(\mathtt{A})|$ by replacing another configuration from the current active set. For this purpose we compute

$$\gamma(\mathrm{cfg}) = \max_{1 \le j \le m} |c_j|, \qquad \text{where}$$
$$\begin{pmatrix} c_1 & \dots & c_m \end{pmatrix} = \begin{pmatrix} b_1 & \dots & b_m \end{pmatrix} \mathtt{A}^{-1}.$$

Thus, if $\gamma(\mathrm{cfg}) > 1$ then $|\det(A)|$ could be increased. In this case $(b_1 \dots b_m)$ should, according to the Maxvol algorithm, replace the row with index

$$k = \mathrm{argmax}_{1 \le j \le m} |c_j|$$

of the matrix $\mathtt{A}$, and the active set should be hence updated.

Note that if the active set and the training set are the same, then the parameters can be found as

$$\xi = \left(E^{\mathrm{qm}}(\mathrm{cfg}_1) \dots E^{\mathrm{qm}}(\mathrm{cfg}_m)\right)\mathtt{A}^{-1}$$

and the energy of any configuration can be expressed as

$$E^{\mathrm{mtp}}(\mathrm{cfg}) = \sum_{j=1}^{m} c_j E^{\mathrm{qm}}(\mathrm{cfg}_j).$$

This mathematical formula allows us to formally say that MTP extrapolates if $\gamma(\mathrm{cfg}) > 1$, and interpolates otherwise. We hence interpret γ as the *extrapolation grade*.

In the context of this formula it is worthwhile to discuss the fact that the computed reference DFT energies $E^{\mathrm{qm}}(\mathrm{cfg}_j)$ (as well as its gradients) have numerical errors due to finite k-points or basis set, which can be interpreted as noise. Furthermore, even if $E^{\mathrm{qm}}(\mathrm{cfg}_j)$ were accurately computed, from the point of view of a local-iteration model (1) the variations in DFT energies arising from different environments of an atom beyond the cutoff distance will also be considered as noise. By limiting $\max_{1 \le j \le m} |c_j|$ above by 1 or some other threshold, we also control the extent to which the noise in $E^{\mathrm{qm}}(\mathrm{cfg}_j)$ (both numerical and the one arising from non-locality of iteratomic interaction) is amplified. Because the prediction $E^{\mathrm{mtp}}(\mathrm{cfg})$ is a linear combination of many such $E^{\mathrm{qm}}(\mathrm{cfg}_j)$, the noise in $E^{\mathrm{mtp}}(\mathrm{cfg})$ could in principle be significantly higher than that of DFT. Therefore, to reduce such noise one may need to choose better convergence parameters than what is normally used for production DFT calculations and/or add sample additional (random) configurations to the training database to reduce the noise by an averaging effect. To the best of our knowledge, this issue has not been properly investigated in the context of machine-learning potentials and could be an important subject of future research[4].

In practical simulations with MLIP we use the so-called two-threshold scheme illustrated in figure 2. In this scheme we choose two thresholds: γ_{select}—threshold beyond which we select a configuration for training, and γ_{break}—threshold beyond which we terminate the simulation. The rationale for this is the following. If γ_{select} is not too high (typically around 2) then for configurations cfg with $\gamma(\mathrm{cfg}) < \gamma_{\mathrm{select}}$ the error of prediction is usually not significantly higher than for interpolative configurations. When $\gamma_{\mathrm{select}} < \gamma(\mathrm{cfg}) < \gamma_{\mathrm{break}} \approx 10$ the error of predictions may be significantly higher, however, the simulation still remains reliable and does not have to be terminated. The values $1.1 \lesssim \gamma_{\mathrm{select}} \lesssim 5$ and $3 \lesssim \gamma_{\mathrm{break}} \lesssim 20$ appear to be universal for any atomistic system that has been tried so far, however, the researchers are advised to do their own testing.

To generalize the D-optimality criterion to nonlinearly parametrized MTP, assume that the values of the parameters, $\bar{\theta}$, are already near the optimal ones and we hence linearize the energies $\mathbf{E}^{\mathrm{mtp}}$ with respect to the parameters. The matrix $\mathtt{B}$ in this case is a tall Jacobi matrix

[4] We thank the anonymous referee for bring our attention to the importance of discussion of the numerical DFT noise.

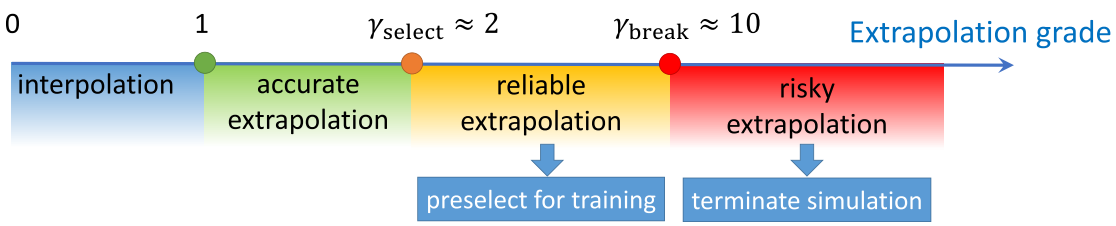


Figure 2. Classification of extrapolation grades. A configuration with the grade less than γ_{select} does not trigger any active learning actions, a configuration whose grade is between γ_{select} and γ_{break} are considered reliable yet useful for extension of the training set, but those with the grade larger than γ_{break} are risky and trigger termination of the simulation.

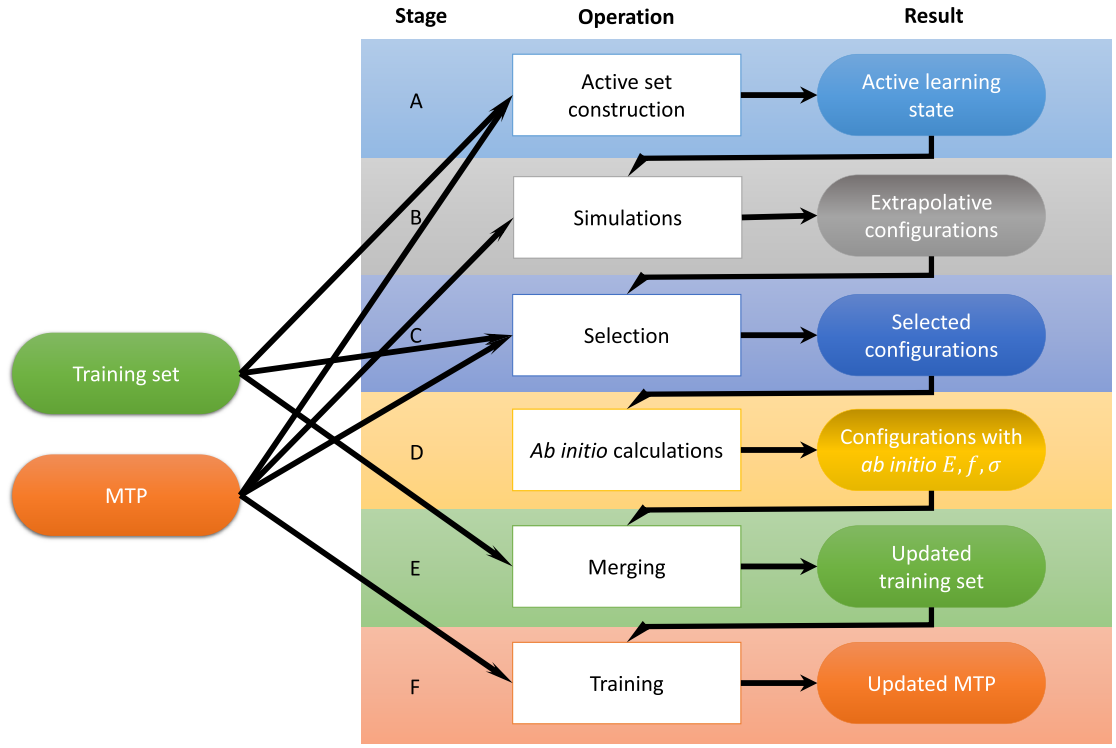


Figure 3. Scheme of active learning bootstrapping iterations.

$$\mathsf{B} = \begin{pmatrix} \frac{\partial}{\partial\theta_1} E^{\text{mtp}}\left(\text{cfg}_1;\bar{\theta}\right) & \dots & \frac{\partial}{\partial\theta_m} E^{\text{mtp}}\left(\text{cfg}_1;\bar{\theta}\right) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial\theta_1} E^{\text{mtp}}\left(\text{cfg}_K;\bar{\theta}\right) & \dots & \frac{\partial}{\partial\theta_m} E^{\text{mtp}}\left(\text{cfg}_K;\bar{\theta}\right) \end{pmatrix},$$

where each row corresponds to a particular configuration from the training set. The matrix A is represented in the analogous manner, and other details of the active learning algorithm remain the same.

2.4. Active learning bootstrapping iterations

As discussed in the previous section, a good way of assembling a training set is by sampling configurations directly from the atomistic simulation we are planning to conduct. This is often called learning on-the-fly, following pre-machine-learning algorithms [63]. However, in the simple learning-on-the-fly strategy there are issues, for instance, with energy conservations after refitting the potential. To address that, we use a bootstrapping technique in which we terminate the simulation, retrain a potential, and restart a simulation. This way, no change in the underlying potential energy surface during a single simulation is committed.

Thus, one iteration of an active learning algorithm consists of the following steps (see figure 3).

(a) On the first step, the active set is selected among all configurations from the training set and the active learning state is formed.
(b) Run the simulation with the current potential and active selection of the extrapolative configurations ($\gamma(\text{cfg}) > \gamma_{\text{select}}$). The simulation is running until its successful completion or until $\gamma(\text{cfg}) \geq \gamma_{\text{break}}$.

(c) If a simulation stopped after exceeding the maximum allowed extrapolation grade ($\gamma(\text{cfg}) \geq \gamma_{\text{break}}$), an update of the active set has to be performed. To that end, the Maxvol algorithm is used to select the new configurations that will be appended to the training set among all extrapolative configurations sampled at the first step.
(d) The selected configurations are calculated with the *ab initio* model.
(e) Next the selected configurations with the *ab initio* the energy, forces, and stresses are appended to the trained set.
(f) The MTP is retrained.

Each iteration of this scheme extends the training region and improves the stability of the MTP (i.e. simulation runs longer without termination by extrapolation). The iterations proceed until the simulation is finished without exceeding the critical value of extrapolation γ_{break}.

3. MLIP package

MLIP is a software package implementing MTP. It is distributed free of charge for non-commercial purposes and can be obtained at [73]. We briefly describe this software package here, and in detail in [71].

The package can be compiled into the library providing interface between MLIP and other packages, most notably LAMMPS [75], and `mlp` binary that provides basic operations including:

`convert-cfg`	converting VASP or LAMMPS input/output files to the internal `.cfg` format of atomic configuration,
`train`	training MTPs on `.cfg` datasets,
`mindist`	computes the minimal distances in configurations,
`calc-efs`	using them to evaluate energy, forces, and stresses on a database,
`calc-grade`	computing extrapolation grades and creating an active learning state for active-learning simulations,
`select-add`	selecting a limited number of configurations from a large set of extrapolative configurations to be added to the training set (the name of command comes from two actions: selecting and adding),
`relax`	use the internal structure relaxation algorithm to relax (i.e. minimize the potential energy of) configurations from a file.

A list and a short description of the commands can be obtained by executing

```
mlp list
```

and

```
mlp help <command>
```

The commands `train` and `relax` work with MPI parallelism, the rest of the commands are serial.

The package contains untrained potentials defining MTPs of level 2, 4, …, 28.

When called from external atomistic simulation codes or the `relax` command, the behavior of MLIP is controlled by the MLIP settings file, most often named `mlip.ini`, described in [71]. There one can specify what MTP file to use, whether active learning should be switched on, the values of extrapolation thresholds, etc.

4. Example 1: elastic constants of molybdenum

In this example we demonstrate how to passively train an MTP, calculate training and validation errors, and calculate energy/volume curve and elastic constants using bcc-Mo as an example. The files referenced in this section are available at [72].

4.1. Training and validation of MTP

We choose the functional form of MTP of level 16 (the file `untrained_mtps/16.mtp` of [73]), eight radial basis functions, and set $R_{\text{cut}} = 5.2$ Å and $R_{\text{min}} = 1.9$ Å. We fit an ensemble of five MTPs in order to estimate the uncertainty of predictions—we will show that our estimated uncertainty reliably predicts the error of MTP as compared to DFT. We take the training and test sets from [62] available at the public git repository[5]. We filter out configurations with the minimal distance between atoms smaller than 1.9 Å. The filtering and checking was facilitated by the `mlp mindist` command. We further recompute both sets of configurations with the VASP package [76–78] with slightly higher DFT convergence parameters, namely, the energy cutoff `ENCUT` = 400 eV and the Γ-centered k-point mesh with `KSPACING` = 0.114. To fit an MTP we run the following command:

[5] https://github.com/materialsvirtuallab/mlearn.

Table 1. Average training and validation errors for the ensemble of five MTPs and their uncertainty estimation with 95% (i.e. 2-sigma) confidence interval. The uncertainty in the errors is small.

	Energy error meV atom^{-1}	Force error meV Å^{-1} (%)	Stress error GPa (%)
Training	5.54 ± 2.10	175 ± 4	0.46 ± 0.04
		$(12.3 \pm 0.4\%)$	$(4.4 \pm 0.4\%)$
Validation	5.10 ± 1.12	180 ± 6	0.46 ± 0.09
		$(12.6 \pm 0.4\%)$	$(4.4 \pm 0.8\%)$

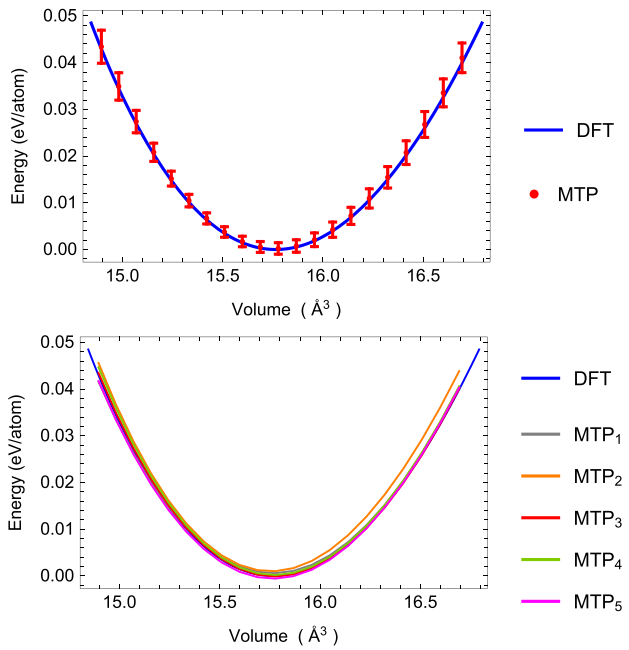


Figure 4. Energy–volume curve for MTP and DFT. An ensemble of five MTPs was used to assess the uncertainty of prediction. Top graph: The MTP error bars correspond to the 95% confidence interval (i.e. 2-sigma interval). Bottom graph: The individual MTPs are plotted against DFT. One can see that the uncertainty is rather small—on the order of a meV/atom—and is indeed centered around the DFT values.

`mlp train init.mtp train.cfg---trained-pot-name = pot.mtp---valid-cfgs = test.cfg`

The `init.mtp` file does not include the fitting parameters, so they are initialized randomly in the beginning of training. The file `train.cfg` contains the training set in the MLIP format. The `valid-cfgs` option enables calculation of validation errors (i.e. the errors calculated on the test set) in the end of training, which otherwise could also be computed with the `mlp calc-errors` command. We have used the default values of the fitting weights $w_e = 1$, $w_f = 0.01$, and $w_s = 0.001$ and the default fitting scaling (12). By repeating the training five times we obtain five potentials with the same functional form but different values of parameters due to different random initialization of the parameters. We refer to these five MTPs as the ensemble of MTPs on which we compute the uncertainty of their predictions. The average training and validation root-mean-square errors, for the ensemble of MTPs and their uncertainty (2-sigma or 95% confidence interval) are shown in table 1. These errors are close to each other, and the uncertainty in the errors is small, indicating reliability of training.

4.2. Energy/Volume curve and elastic constants

For the calculation of elastic constants we first find the equilibrium lattice constant by finding the minimum of the energy/volume curve. To that end, we generate the file `deformed.cfg` with bcc-Mo configurations with compressed/stretched lattice constant. We calculate the energies of these configurations by executing

`mlp calc-efs pot.mtp deformed.cfg deformed_efs.cfg`

The configurations with the energies calculated are written to `deformed_efs.cfg`. The energy volume/curve is shown in figure 4. The calculated MTP lattice constant together with its uncertainty is 3.15928 ± 0.00097 Å (95% confidence interval computed on the ensemble of five potentials). The reference DFT lattice constant is 3.15918 Å—well within the MTP confidence interval.

Table 2. Elastic constants of bcc-Mo as calculated by MTP and DFT. The 95% confidence interval was given for the uncertainty of evaluating the elastic constants by MTP and well encloses the reference DFT results.

	C_{11} (GPa)	C_{12} (GPa)	C_{44} (GPa)
MTP	466 ± 22	161 ± 20	98 ± 14
DFT	472	154	95

We next calculate the elastic constants C_{11}, C_{12}, and C_{44} of bcc-Mo using the finite difference method by applying $\pm 2\%$ strain to individual unit cell components. We prepare the corresponding configurations in `.cfg` files and evaluate their stresses with the `calc-efs` command for MTP, and VASP calculations for DFT, with the same parameters as described above.

The elastic constants calculated with the ensemble of MTPs and using DFT are given in table 2. As could be seen, the uncertainty of the calculation is relatively small, the MTP and DFT constants are close to each other, and the DFT constants fall well within the 95% confidence interval of MTP.

We cross-check our finite-difference results using the LAMMPS script `in.elastic`, adopted from the corresponding script from the `examples/ELASTIC/` folder of LAMMPS. To run the script with MTP, one must simply declare the MTP `pair_style` potential in the script:

```
pair_style mlip mlip.ini pair_coeff * *
```

where `mlip.ini` is the file with the MLIP settings indicating that `pot.mtp` should be used. The script directly yields the elastic constants of bcc-Mo computed with the trained MTP. The difference between the LAMMPS and finite-difference results are insignificant, between 0 and 2 GPa.

5. Example 2: melting point of aluminum

Here we describe how to compute the melting point of aluminum by actively training an MTP on-the-fly. The choice of Aluminum as a benchmark system is because of the availability of extremely accurate DFT results [79]. The detailed description of the input, output, and intermediate files are given in the `example-2/README` file of [72] and files referenced therein.

5.1. Active learning iterations as implemented in MLIP

We start with the description of the way the active learning iterations (section 2.4) are implemented in MLIP, following figure 3.

Step A active learning state preparation: The iteration starts by generating the `state.als` file based on the training set. This is done by executing

```
mlp calc-grade pot.mtp train.cfg train.cfg temp.cfg
```

This creates the `state.als` file containing the matrices described in section 2.3 which are needed for running LAMMPS with active learning. (The file `temp.cfg` can be discarded.)

Step B simulations: Now we use LAMMPS to run an MD simulation, switching on the active selection of configurations as indicated in the `mlip.ini` file. The extrapolative configurations are written to `preselected.cfg`. According to the thresholds set in `mlip.ini`.

Step C selection: There may be a lot of configurations in `preselected.cfg`, especially when multiple MD trajectories are generated at Step B—in the later case their extrapolative configurations are merged into a single `preselected.cfg` file. To select only non-repetitive, representative configurations, we use the `select-add` command:

```
mlp select-add pot.mtp train.cfg preselected.cfg selected.cfg
```

This command forms the matrix (13) and selects those configurations that maximize the determinant as explained in section 2.3. The total number of selected configurations is guaranteed to be less than the number of parameters in `pot.mtp` (usually up to a few hundred) and is independent of the total number of configurations in `preselected.cfg`.

Step D *ab initio* calculations: Now we calculate DFT energies, forces, and stresses for `selected.cfg` and write them to a new file, `computed.cfg`. In principle, this file may contain only a subset of configurations same as or close to the ones in `selected.cfg`—e.g. in the case if some DFT calculations were not done due to convergence issues or technical problems. In this case missing configurations will be selected at the next iteration[6].

[6] The configurations at the next iterations will be different from the ones selected at the current iteration, due to the fact that the potential will be changed. In some applications this helps to solve issues arising from the lack of convergence of DFT self-consistent iterations, e.g. in cases when a configuration is at a cross-over between two magnetic states—if a configuration was at the cross-over, at the text iteration the selected configuration will be at a slightly changed.

Step E merge: We append `computed.cfg` to `train.cfg` from the previous iteration. We consider this a separate step only to simplify the data flowchart in figure 3.

Step F training: Finally, we re-fit the potential on the updated training set. The produced potential, together with the expanded training set are the output of the iteration. Considering all other files as intermediate ones, the net result of the iteration is the expanded training set and an updated potential that is able to make predictions in a larger configurational space.

5.2. Stage 1: active learning of low-fidelity DFT during MD

Our first goal is to construct a potential trained on low-fidelity DFT calculations, whose purpose would be to be able to robustly sample configurations from solid and liquid phase of Al. This is achieved in Stage 1.

Instead of starting the active learning cycle with an empty training set, we first generate an initial training set by running a 90 fs long VASP MD trajectory, adding every tenth configuration to the training set to avoid correlated configurations and save it to the file `train.cfg`. We run an NVT-MD starting from an ideal 108-atom fcc configuration with the lattice parameter of 4.1 Å. Only Γ-point was used for the k-point integration and the energy cutoff of 410 eV was used for the plane-wave basis. The training set, thus, contains ten configurations.

We choose the potential of level 16 with cutoff of 5 Å, `mindist` of 2 Å, and the radial basis size of 6, and save it to `init.mtp`. We next train this potential on the database `train.cfg` with the following command:

```
mlp train init.mtp train.cfg---trained-pot-name = pot.mtp
```

The potential `pot.mtp` and training set `train.cfg` are the input to the active learning iteration.

5.2.1. Stage 1a: active learning on a single MD trajectory

We now run active learning iterations, in which we run molecular dynamics and collect configurations on which the potential attempts to extrapolate, in accordance to the general workflow shown in figure 3 and detailed in section 5.1 focusing on data processing aspect of an active learning iteration. We set the thresholds $\gamma_{\text{select}} = 3$ and $\gamma_{\text{break}} = 10$. We now describe the steps of the active learning iteration.

We actively train an MTP while running an NVT-MD under the same setting as the initial AIMD trajectory—at 900 K starting from a 108-atom fcc configuration with the lattice parameter of 4.1 Å. The initial velocities were kept the same in order to ensure that the MD trajectory on subsequent iterations are close to those on the previous iteration—which in turn was done solely for illustration purposes to see the gradual increase in the number of time steps an MD with MTP can reliably do.

During the first iteration the MTP successfully recognizes that the initial configuration is in the training set, makes a time step and selects the next configuration to be added to the training set. (LAMMPS and VASP initialize the MD with different velocities, hence the trajectories and different.) On the second iteration, four time steps were made until the simulation exceeded the upper threshold, and two configurations (sampled from the last two time steps) were added to the training set. This process continued until the 58th iteration, on which a 10-ps MD was run and no configuration was selected as extrapolative. Totally, 225 configurations were actively selected during the entire process by the 58th iteration.

To better understand the performance of the active learning iterations, we generated a validation set of 200 configurations sampled at 900 K and computed them with DFT. We emphasize that this validation set is typically not needed in practice in the active learning iterations other than for testing purposes. The validation error and the number of time steps of MD at each iteration is plotted in figure 5. The number of time steps is a good indicator of the reliability of the potential. One can see that even a potential trained on the first 10 configurations has a good validation error of 13.5% which in the course of active training goes down to only 12%. The ten configurations have altogether 1080 forces which appears to be sufficient for reaching a good accuracy on a potential with about 120 parameters. However, the algorithm requires extra 225 configurations to attain the perfect reliability of the simulation.

The resulting 235 configurations are rather correlated—they were collected from essentially the same MD trajectory approximately every 10 time steps. Hence on the 59th iteration we sparsify the training set: we select only those configurations with respect to which other configurations are interpolative. We do it with the `select-add` command. It selects 78 configuration that we keep for the subsequent iterations.

5.2.2. Stage 1b: active learning during multi-scenario MD simulations

Next we perform the 60th iteration in the same manner as iterations 1–58 except that we run 60 LAMMPS trajectories in parallel, with the lattice parameter a between 4.0 and 4.25 Å and temperatures T between 700 and 1100 K for two scenarios: `fcc` and `liquid`. The `fcc` scenario is exactly the same as described above, and in the liquid scenario we first run 10 000 steps with $a = 4.25$ Å and $T = 1500$ K, then we run 10 000 steps with the target temperature (between 700 and 1100 K) gradually reducing a to the target value, and finally run 10 000 steps with the target temperature and lattice parameter. The `preselected.cfg` files from these

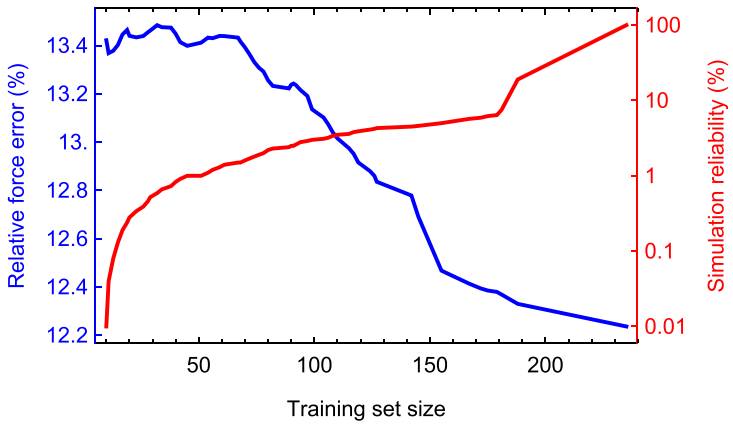


Figure 5. Performance of active learning bootstrapping iterations on Stage 1a. The reliability is defined as the number of time steps made before exceeding γ_{break} relative to the total number of time steps (10 000). One can see that the error drops only slightly—from 13.5% to 12%. As will be shown, the large error is mostly due to the k-point noise in the data that will be removed on the text stage. The major effect of the iterations is reliability—it increases from 0% to 100% after reaching 235 configurations in the training set.

trajectories are concatenated into one file before doing the selection step. The thresholds are set $\gamma_{\text{select}} = \gamma_{\text{break}} = 5$. Equal thresholds ensure that only one configuration will be selected from each trajectory—which will ensure that not too many configurations will be given to the `select-add` command.

Again, we have generated a validation set of 180 configurations corresponding to the above 60 MD scenarios, in order to understand how the active learning iterations perform. Before the 60th iteration the measured force error was 20%.

This was repeated for 14 additional iterations, until at the 74th iteration no configurations were selected for training. The size of the training set reached 202 configurations by the 75th iteration, which means that total 359 static DFT calculations were made. We note that because only the Γ-point was used, these 359 were rather cheap and did not take the majority of computational time during this process. The force error decreased from 20% to 17%. The performance over iterations 60–74 is plotted on figure 6.

Finally, on the 75th iteration we use the trained potential to sample 186 solid and liquid configurations that we later compute with high-accuracy DFT. To that end, we switch off active learning (option `select FALSE` in the `mlip.ini` file) in the LAMMPS simulations. This set was generated similarly to the validation set, but with extra six configurations corresponding to liquid at $T = 1500$ K and $a = 4.25$—we use these parameters to prepare the liquid state in all liquid simulations. It has been observed that sampling configurations from MD in addition to active sampling improves the accuracy of the potential, however, it has not been investigated to date whether this is related to different probability distributions of MD configurations and actively sampled configurations, or it is related to the noise averaging as discussed in section 2.3.

5.3. Stage 2: high-accuracy DFT

This stage starts with 186 liquid and solid configurations computed with the $3 \times 3 \times 3$ k-points mesh. We fit five MTPs to this training set, with random initialization of parameters, and select the one with the lowest error—this is the beginning of the new round of active learning iterations. We use the same 180-configuration validation set computed with the new DFT parameters to assess the error. The error immediately drops from 17% to 4.5%—this is the result of improving the accuracy of the data. This means that on Stage 1 the magnitude of the error was high mostly due to the k-point noise in data.

We next run the same iterations as in Stage 1b: the only difference is a more accurate k-point mesh in the DFT calculations on Step D. We repeat the active learning simulations until we obtain a potential with which we run a molecular dynamics and select no extrapolative configurations. We use this potential for the last stage.

5.4. Stage 3: coexistence simulations

At this stage we completely switch off active learning and perform the standard coexistence simulations with LAMMPS. We start by running four molecular dynamics simulation of solid and liquid with $4 \times 12^3 \approx 7000$ atoms in a supercell with $T \in \{900\text{ K}, 1000\text{ K}\}$. Without active learning, LAMMPS with MLIP can be run in the MPI-parallel mode, efficiently accelerating the simulation of 7000 atoms. We use them to fit the fcc lattice

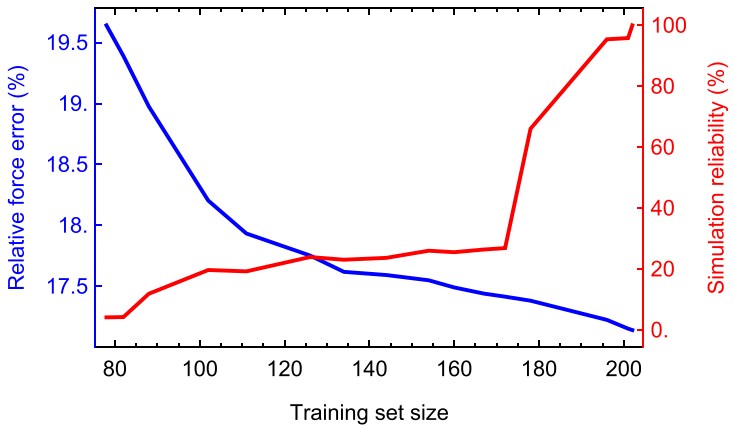


Figure 6. Performance of active learning bootstrapping iterations on Stage 1b. The reliability is defined as the number of time steps made before exceeding γ_{break} relative to the total number of time steps. As in Stage 1a, the large error is mostly due to the k-point noise in the data that will be removed on the text stage. Likewise, the major effect of the iterations is reliability—it increases from 5% to 100% after reaching 202 configurations in the training set.

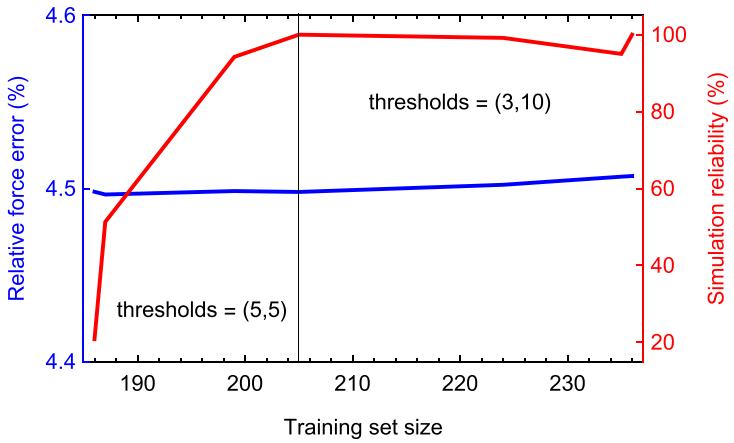


Figure 7. Performance of active learning bootstrapping iterations on Stage 2. The reliability is defined as the number of time steps made before exceeding γ_{break} relative to the total number of time steps. The slight decline in reliability is because of how it is measured—γ_{break} was reduced to a tighter value after reaching 205 configurations in the training set. The error almost does not change. As before, the major effect of the iterations is reliability.

parameter as a function of temperature, and the coefficient of expansion of the 'long axis' for the 50%–50% solid-liquid coexistence system.

We then set up a coexistence simulations in the following way: (0) we start by a $12 \times 12 \times 48$ system as measured in the lattice parameter units. We run 3000 steps of the NVT with the target temperature T_{target} and the corresponding lattice parameter, then we freeze the lower half ($12 \times 12 \times 24$) of the system and run an MD simulation at 1200 K, expanding the supercell in the third axis until the supercell volume will correspond to the target temperature. Finally, we reset the velocities of all the atoms to a lower temperature T_{reset}, to compensate for the higher energy of the liquid, and run a classical MD with the NVE ensemble. After a few runs we found that $T_{\text{target}} = 885$ K and $T_{\text{reset}} = 610$ K yields a 50%–50% coexistence, with the axial stresses less than 0.02 GPa by absolute value and coexistence temperature of 885 K. In other words, the computed melting temperature of Al on the $3 \times 3 \times 3$ k-point mesh is 885 K, just 3 K smaller than the extremely accurate DFT calculation of [79].

6. Example 3: stable convex hull of Ag-Pd structures

In the last example we show how to calculate the convex hull of the Ag-Pd binary system using active learning. Constructing a convex hull means identifying the most stable (with the lowest formation energy) binary structures with the composition $Ag_{1-x}Pd_x$ with x between 0 (pure Ag) and 1 (pure Pd). We identify the stable structures by selecting among the finite number of the so-called 'candidate structures' those ones

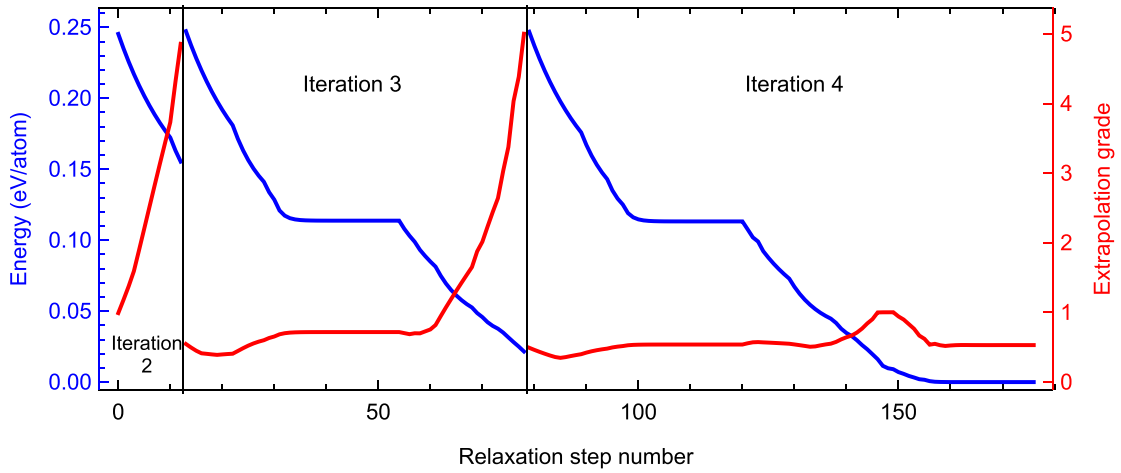


Figure 8. Evolution of the extrapolation grade and the energy above the equilibrium of a structure during three iterations of the active learning algorithm. On iteration 1 (not plotted) the structure is selected to the training set without making a single relaxation step. On iteration 2 the extrapolation grade starts with 1 but quickly reaches $\gamma_{\text{select}} = \gamma_{\text{break}} = 5$ within about 12 relaxation steps. On iteration 3 the relaxation algorithm makes better progress but still ends up by encountering an extrapolative structure, while iteration 4 finishes successfully.

that lie on the lower convex hull on the energy-composition plot. As for the previous examples, the files mentioned below are available at [72].

The structures themselves may be of different origin, but usually they are provided by some generative algorithm (e.g. [80]) or are taken from some bank of structures (e.g, [81], [82]). We follow the second way: generate 39k crystal structures with up to 12 atoms and different underlying lattice types (fcc, bcc, hcp) populated with different numbers of Ag and Pd atoms to introduce different concentrations.

The samples then undergo structure relaxation (i.e. energy minimization) to relieve interatomic forces and lattice stresses by changing (relaxing) lattice vectors and atomic positions. Similarly to the MD simulation from section 5.1, each relaxation produces a trajectory starting with the structure to be relaxed and ending with the 'relaxed' one having practically zero forces and stresses. The active learning was used in this scenario as well, with the source of new configurations being the relaxation trajectories of candidate structures.

To construct a convex hull for the Ag-Pd system (simultaneously fitting a corresponding moment tensor potential) we launch the active learning procedure as described in section 5.1, this time starting with an empty training set.

The only essential difference of the active learning workflow compared to section 5.1 is that the step B (simulation) is replaced by relaxation with the MLIP package. During this step we use MLIP to run relaxation for each candidate structure. Each relaxation can end up in two ways: it can either finish successfully producing an equilibrium configuration; or terminate early if an extrapolative configuration occurs in the relaxation trajectory. The successfully relaxed structures are appended to the file `relaxed.cfg`. Active learning iterations keep going until all the relaxation iterations have finished successfully providing the corresponding equilibrium structures. After the iterations end, the last iteration produces the `relaxed.cfg` file containing all the configurations successfully relaxed. Based on the energies of the relaxed structures provided by MTP, their formation energies are calculated, which allows then to construct a convex hull (see figure 9).

To get an insight into the evolution of the extrapolation grade of a structure across iterations, we choose a structure from the pool of 39k structures and plot its energy above the corresponding equilibrium structure and the extrapolation as the function of the relaxation step number. The graphs are shown in figure 8.

To compare the performance of different MTPs the convex hull for AgPd system was constructed using potentials of three levels, 12, 16, and 20, each potential ran independently to observe convergence with respect to the level. The errors and the training set size of the three potentials after the last active learning iteration are given in table 3. One can see that through the level of the potential we can control the accuracy and the training set size. The convex hulls obtained by these potentials are shown in figure 9 along with the convex hull based on 302 candidate structures from the Aflow library [82] relaxed with DFT (as implemented in VASP 5.4.4). The effect of nonzero fitting error is that the ground-state structures may slightly miss the MTP-based convex hull as they may rise above other structures in energy—for instance Ag_2Pd on the level-20 potential is 1 meV atom^{-1} above the convex hull. This is nevertheless well within the

Table 3. The mean absolute error (MAE), root mean square error (RMSE), and training set size actively selected while training the potentials of levels 12, 16, and 20 as they relax the AgPd candidate structures. Starting with a potential of higher level results in more DFT calculations (train set size), but yields better accuracy.

lev_{max}	Train set size	Energy MAE (meV atom^{-1})	Energy RMSE (meV atom^{-1})
12	226	2.8	3.8
16	442	1.9	2.4
20	920	1.3	1.7

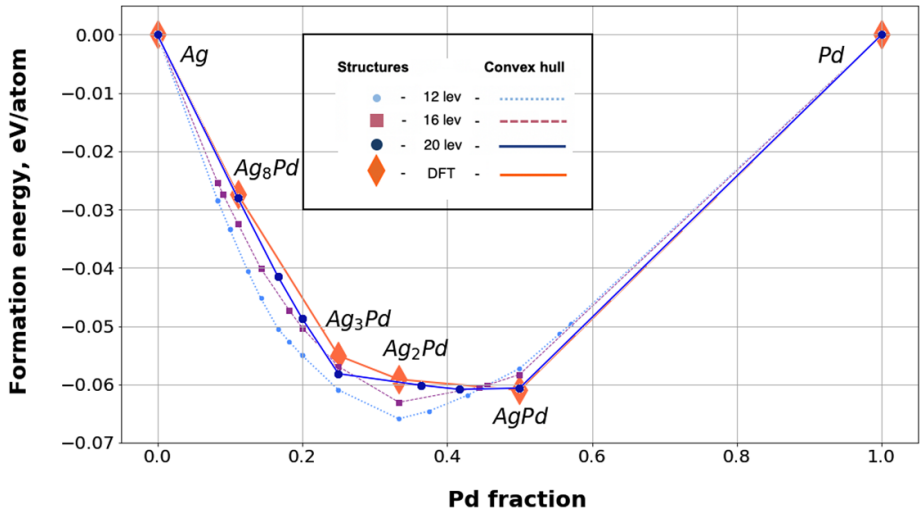


Figure 9. Convex hulls obtained by MTPs of different levels include more structures compared to the one obtained by DFT, thanks to 100 times more candidate structures used. Due to the fitting errors, some DFT ground-state structures are slightly (<1 meV atom^{-1}) above the MTP convex hull level.

range of the fitting error of the potential. On the other hand, some extra structures, not included in the DFT convex hull, appear on the MTP convex hull due to the better coverage of the compositional space by the 39k initial structures used by MTP. As seen from figure 9 using an MTP with more parameters (and hence with more training data) one can approximate the DFT-based convex hull better. If one needs to completely eliminate the MTP errors from the resulting convex hull, one may post-relax some of the most stable structures on DFT (as was done in [25]).

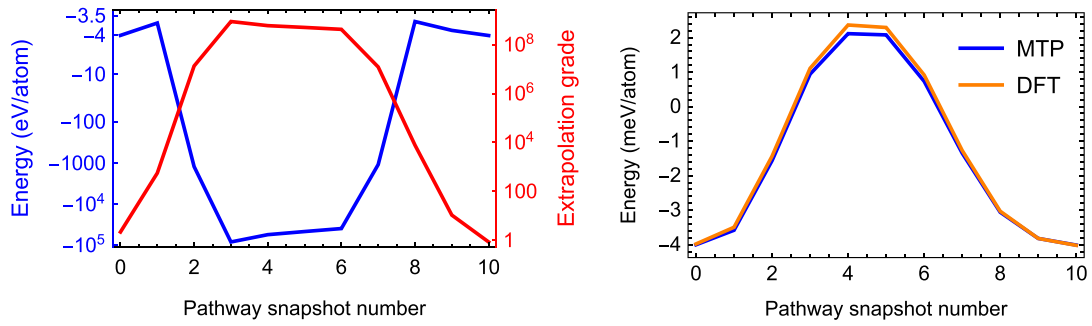
In this example the MTP shows its predictive power, as it has almost precisely reproduced the results obtained through DFT relaxation of very carefully chosen samples, but has fulfilled this task by relaxing a general system-agnostic pool of candidate structures. The benefits from using an active learning approach consist of a several-fold savings in DFT calculations—several hundreds static calculations are much cheaper than 302 relaxations especially considering that these calculations are typically restarted several times to avoid the error of originating from the plane-wave basis being fixed during the DFT relaxation. The computational cost of training and relaxing of the 39k samples is less than 10% and the error of 2 meV atom^{-1} is comparable to discrepancies in formation energies caused by different DFT pseudopotentials, or even different convergence parameters.

6.1. Transition pathway test

To further illustrate the performance of the MLIP software, we revisit the test of predicting the pathway between two stable AgPd structures [83]. The two structures and the pathway were taken to be precisely the same as in [83] and recomputed with our DFT settings. The pathway consists of 11 structures including the starting structure (numbered 0) and the final structure (numbered 10).

We first compute the energy of the pathway structures with the final MTP trained while relaxing the ground-state structure candidates. In the middle of the pathway, the structures are very different from the training set. It is therefore not surprising that the predictions of MTP in the middle of the pathway are completely inadequate when used without active learning, as seen in figure 10(a). Nevertheless, near the two ends of the pathway, the MTP correctly predicts the increase in energy, without which MTP would not be able to relax correctly those equilibrium structures corresponding to the two ends of the pathway.

We consider it important to emphasize that we view it to be the wrong use of MLIP to evaluate configurations that we know are far from the training set without active learning. Indeed, when active



(a) MTP-predicted pathway before active learning

(b) DFT pathway and MTP prediction after active learning

Figure 10. The MTP trained on ground-state structures inadequately predicts the energy profile of the pathway between the two ground states, nevertheless, the MTP is 'aware' through its extrapolation grade that these predictions cannot be trusted. After actively learning the pathway, MTP can accurately predict its energy profile.

learning is switched on, the MTP becomes 'aware' through its extrapolation grade (red curve in figure 10(a)) that these predictions cannot be trusted. Active learning, when applied to the pathway structures, selects all 11 of them for training. Once trained, MTP predicts very well the pathway energy profile between the two structures, as seen in figure 10(b). Because all 11 structures are in the training set, their extrapolation grades are all equal to 1, thus indicating that the predictions are now reliable.

7. Concluding remarks

This manuscript gives a formulation of the MTPs and the active learning algorithm, outlines the structure of the MLIP code implementing MTPs and active learning, and provides three detailed examples of the usage of MLIP to perform particular atomistic simulations. Our goal was that the simulations could be easily reproduced, therefore the examples were chosen to be sufficiently familiar for practitioners, yet advanced enough to contain many of the components of simulations that the frontiers of computational materials science need. The supplementary material [72] contains all the files and data needed to reproduce the results presented in the manuscript.

Acknowledgment

This work was supported by the Russian Science Foundation (Grant No. 18-13-00479).

ORCID iD

Alexander V Shapeev https://orcid.org/0000-0002-7497-5594

References

[1] Behler J and Parrinello M 2007 Generalized neural-network representation of high-dimensional potential-energy surfaces *Phys. Rev. Lett.* **98** 146401
[2] Artrith N and Behler J 2012 High-dimensional neural network potentials for metal surfaces: a prototype study for copper *Phys. Rev.* B **85** 045439
[3] Behler J 2016 Perspective: machine learning potentials for atomistic simulations *J. Chem. Phys.* **145** 170901
[4] Dolgirev P E, Kruglov I A and Oganov A R 2016 Machine learning scheme for fast extraction of chemically interpretable interatomic potentials *AIP Adv.* **6** 085318
[5] Gastegger M and Marquetand P 2015 High-dimensional neural network potentials for organic reactions and an improved training algorithm *J. Chem. Theory Comput.* **11** 2187–98
[6] Purja Pun G P, Batra R, Ramprasad R and Mishin Y 2019 Physically informed artificial neural networks for atomistic modeling of materials *Nat. Commun.* **10** 1–10
[7] Smith J S, Isayev O and Roitberg A E 2017 ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost *Chem. Sci.* **8** 3192–203
[8] Zhang L, Han J, Wang H, Car R and Weinan E 2018 Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics *Phys. Rev. Lett.* **120** 143001
[9] Lubbers N, Smith J S and Barros K 2018 Hierarchical modeling of molecular energies using a deep neural network *J. Chem. Phys.* **148** 241715

[10] Schütt K, Kindermans P-J, Sauceda Felix H E, Chmiela S, Tkatchenko A and Müller K-R 2017 Schnet: A continuous-filter convolutional neural network for modeling quantum interactions *Advances in Neural Information Processing Systems* pp 991–1001 (https://papers.nips.cc/paper/2017/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf)
[11] Bartók A P, Payne M C, Kondor R and Gábor C 2010 Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons *Phys. Rev. Lett.* **104** 136403
[12] Deringer V L and Gábor C 2017 Machine learning based interatomic potential for amorphous carbon *Phys. Rev.* B **95** 094203
[13] Grisafi A, Wilkins D M, Csányi G and Ceriotti M 2018 Symmetry-adapted machine learning for tensorial properties of atomistic systems *Phys. Rev. Lett.* **120** 036002
[14] Jinnouchi R, Karsai F and Kresse G 2019 On-the-fly machine learning force field generation: application to melting points *Phys. Rev.* B **100** 014105
[15] Szlachta W J, Bartók A P and Gábor C 2014 Accuracy and transferability of gaussian approximation potential models for tungsten *Phys. Rev.* B **90** 104108
[16] Bartók A P, Kondor R and Gábor C 2013 On representing chemical environments *Phys. Rev.* B **87** 184115
[17] Pozdnyakov S N, Willatt M J, Bartók A P, Ortner C, Csányi G and Ceriotti M 2020 On the completeness of atomic structure representations (arXiv: 2001.11696)
[18] Vandermause J, Torrisi S B, Batzner S, Xie Y, Sun L, Kolpak A M and Kozinsky B 2020 On-the-fly active learning of interpretable Bayesian force fields for atomistic rare events *npj Comput. Mater.* **6** 20
[19] Thompson A P, Swiler L P, Trott C R, Foiles S M and Tucker G J 2015 Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials *J. Comput. Phys.* **285** 316–30
[20] Wood M A and Thompson A P 2018 Extending the accuracy of the snap interatomic potential form *J. Chem. Phys.* **148** 241721
[21] Cusentino M A, Wood M A and Thompson A P 2020 Explicit multi-element extension of the spectral neighbor analysis potential for chemically complex systems *J. Phys. Chem.* A **124** 5456–64
[22] Drautz R 2019 Atomic cluster expansion for accurate and transferable interatomic potentials *Phys. Rev.* B **99** 014104
[23] van der Oord C, Dusson Gève, Csányi G and Ortner C 2020 Regularised atomic body-ordered permutation-invariant polynomials for the construction of interatomic potentials *Mach. Learn.: Sci. Technol.* **1** 015004
[24] Shapeev A V 2016 Moment tensor potentials: A class of systematically improvable interatomic potentials *Multiscale Model. Simul.* **14** 1153–73
[25] Gubaev K, Podryabinkin E V, Hart G L W and Shapeev A V 2019 Accelerating high-throughput searches for new alloys with active learning of interatomic potentials *Comput. Mater. Sci.* **156** 148–56
[26] Gubaev K, Podryabinkin E V and Shapeev A V 2018 Machine learning of molecular properties: locality and active learning *J. Chem. Phys.* **148** 241727
[27] Lomaka A and Tamm T 2020 Linearization of moment tensor potentials for multicomponent systems with a preliminary assessment for short-range interaction energy in water dimer and trimer *J. Chem. Phys.* **152** 164115
[28] Hernandez A, Balasubramanian A, Yuan F, Mason S A M and Mueller T 2019 Fast, accurate and transferable many-body interatomic potentials by symbolic regression *npj Comput. Mater.* **5** 1–11
[29] Botu V and Ramprasad R 2015 Learning scheme to predict atomic forces and accelerate materials simulations *Phys. Rev.* B **92** 094306
[30] Zhenwei Li, Kermode J R and Alessandro D V 2015 Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces *Phys. Rev. Lett.* **114** 096405
[31] Chmiela S, Tkatchenko A, Sauceda H E, Poltavsky I, Schütt K T and Klaus-Robert M 2017 Machine learning of accurate energy-conserving molecular force fields *Sci. Adv.* **3** e1603015
[32] Manzhos S, Dawes R and Carrington T 2015 Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces *Int. J. Quantum Chem.* **115** 1012–20
[33] Kostiuchenko T, Körmann F, Neugebauer J and Shapeev A 2019 Impact of lattice relaxations on phase transitions in a high-entropy alloy studied by machine-learning potentials *npj Comput. Mater.* **5** 1–7
[34] Shapeev A V 2017 Accurate representation of formation energies of crystalline alloys with many components *Comput. Mater. Sci.* **139** 26–30
[35] Christensen A S, Bratholm L A, Faber F A and von Lilienfeld O A 2020 FCHL revisited: faster and more accurate quantum machine learning *J. Chem. Phys.* **152** 044107
[36] Sandip D, Bartók A P, Csányi G and Ceriotti M 2016 Comparing molecules and solids across structural and alchemical space *Phys. Chem. Chem. Phys.* **18** 13754–69
[37] Faber F A *et al* 2017 Prediction errors of molecular machine learning models lower than hybrid DFT error *J. Chem. Theory Comput.* **13** 5255–64
[38] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 Neural message passing for quantum chemistry (arXiv: 1704.01212)
[39] Hansen K, Biegler F, Ramakrishnan R, Pronobis W, Von Lilienfeld O A, Müller K-R and Tkatchenko A 2015 Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space *J. Phys. Chem. Lett.* **6** 2326–31
[40] Huang B and Von Lilienfeld O A 2016 Communication: understanding molecular representations in machine learning: The role of uniqueness and target similarity *J. Chem. Phys.* **145** 161102-1-161102-6
[41] Rupp M, Tkatchenko A, Müller K-R and Von Lilienfeld O A 2012 Fast and accurate modeling of molecular atomization energies with machine learning *Phys. Rev. Lett.* **108** 058301
[42] Schütt K T, Arbabzadah F, Chmiela S, Müller K R and Tkatchenko A 2017 Quantum-chemical insights from deep tensor neural networks *Nat. Commun.* **8** 13890
[43] Snyder J C, Rupp M, Hansen K, Müller K-R and Burke K 2012 Finding density functionals with machine learning *Phys. Rev. Lett.* **108** 253002
[44] von Lilienfeld O A, Müller K-R and Tkatchenko A 2020 Exploring chemical compound space with quantum-based machine learning *Nat. Rev. Chem.* **4** 347–58
[45] Podryabinkin E V and Shapeev A V 2017 Active learning of linearly parametrized interatomic potentials *Comput. Mater. Sci.* **140** 171–80
[46] Novoselov I I, Yanilkin A V, Shapeev A V and Podryabinkin E V 2019 Moment tensor potentials as a promising tool to study diffusion processes *Comput. Mater. Sci.* **165** 46–56

[47] Ladygin V V, Korotaev P Y, Yanilkin A V and Shapeev A V 2020 Lattice dynamics simulation using machine learning interatomic potentials *Comput. Mater. Sci.* **172** 109333
[48] Podryabinkin E V, Tikhonov E V, Shapeev A V and Oganov A R 2019 Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning *Phys. Rev.* B **99** 064114
[49] Grabowski B, Ikeda Y, Srinivasan P, Körmann F, Freysoldt C, Duff A I, Shapeev A and Neugebauer J 2019 *Ab initio* vibrational free energies including anharmonicity for multicomponent alloys *npj Comput. Mater.* **5** 1–6
[50] Jafary-Zadeh M, Khoo K H, Laskowski R, Branicio P S and Shapeev A V 2019 Applying a machine learning interatomic potential to unravel the effects of local lattice distortion on the elastic properties of multi-principal element alloys *J. Alloys Compd.* **803** 1054–62
[51] Korotaev P Y, Novoselov I I, Yanilkin A V and Shapeev A V 2019 Accessing thermal conductivity of complex compounds by machine learning interatomic potentials *Phys. Rev.* B **100** 144308
[52] Wang C, Aoyagi K, Wisesa P and Mueller T 2020 Lithium ion conduction in cathode coating materials from on-the-fly machine learning *Chem. Mater.* **32** 3741–52
[53] Novikov I S and Shapeev A V 2019 Improving accuracy of interatomic potentials: more physics or more data? a case study of silica *Mater. Today Commun.* **18** 74–80
[54] Novikov I S, Shapeev A V and Suleimanov Y V 2019 Ring polymer molecular dynamics and active learning of moment tensor potential for gas-phase barrierless reactions: application to S + H2 *J. Chem. Phys.* **151** 224105
[55] Novikov I S, Suleimanov Y V and Shapeev A V 2018 Automated calculation of thermal rate coefficients using ring polymer molecular dynamics and machine-learning interatomic potentials with active learning *Phys. Chem. Chem. Phys.* **20** 29503–12
[56] Mortazavi B, Novikov I S, Podryabinkin E V, Roche S, Rabczuk T, Shapeev A V and Zhuang X 2020 Exploring phononic properties of two-dimensional materials using machine learning interatomic potentials *Appl. Mater. Today* **20** 100685
[57] Mortazavi B, Podryabinkin E V, Novikov I S, Roche S, Rabczuk T, Zhuang X and Shapeev A V 2020 Efficient machine-learning based interatomic potentials for exploring thermal conductivity in two-dimensional materials *J. Phys.: Mater.* **3** 02LT02
[58] Mortazavi B, Shojaei F, Shahrokhi M, Azizi M, Rabczuk T, Shapeev A V and Zhuang X 2020 Nanoporous C3N4, C3N5 and C3N6 nanosheets; novel strong semiconductors with low thermal conductivities and appealing optical/electronic properties *Carbon* **167** 40–50
[59] Raeisi M, Mortazavi B, Podryabinkin E V, Shojaei F, Zhuang X and Shapeev A V 2020 High thermal conductivity in semiconducting janus and non-janus diamanes *Carbon* **167** 51–61
[60] Mortazavi B, Roche S, Rabczuk T, Zhuang X and Shapeev A V *et al* 2020 Machine learning interatomic potentials enable first-principles multiscale modeling of lattice thermal conductivity in graphene/borophene heterostructures *Mater. Horizons* **7** 2359–67
[61] Nyshadham C, Rupp M, Bekker B, Shapeev A V, Mueller T, Rosenbrock C W, Csányi G, Wingate D W and Hart G L W 2019 Machine-learned multi-system surrogate models for materials prediction *npj Comput. Mater.* **5** 51
[62] Zuo Y *et al* 2020 Performance and cost assessment of machine learning interatomic potentials *J. Phys. Chem.* A **124** 731–45
[63] Csányi G, Albaret T, Payne M C and Vita A D 2004 'Learn on the fly': a hybrid classical and quantum-mechanical molecular dynamics simulation *Phys. Rev. Lett.* **93** 175503
[64] De Vita A and Car R 1997 A novel scheme for accurate MD simulations of large systems *MRS Online Proc. Library Archive* vol 491 pp 473–80
[65] Settles B 2012 Active learning *Synthesis Lectures on Artificial Intelligence and Machine Learning* vol 6 pp 1–114 (https://www.morganclaypool.com/toc/aim/1/1)
[66] Zhang L, Lin D-Y, Wang H, Car R and Weinan E 2019 Active learning of uniformly accurate interatomic potentials for materials simulation *Phys. Rev. Mater.* **3** 023804
[67] Smith J S, Nebgen B, Lubbers N, Isayev O and Roitberg A E 2018 Less is more: sampling chemical space with active learning *J. Chem. Phys.* **148** 241733
[68] Botu V and Ramprasad R 2015 Adaptive machine learning framework to accelerate *ab initio* molecular dynamics *Int. J. Quantum Chem.* **115** 1074–83
[69] Bernstein N, Csányi G and Deringer V L 2019 De novo exploration and self-guided learning of potential-energy surfaces *npj Comput. Mater.* **5** 1–9
[70] Sivaraman G, Krishnamoorthy A N, Baur M, Holm C, Stan M, Csányi G, Benmore C and Álvaro V-M 2019 Machine learning inter-atomic potentials generation driven by active learning: a case study for amorphous and liquid hafnium dioxide (arXiv: 2001.11696)
[71] Novikov I S, Gubaev K, Podryabinkin E V and Shapeev A V Supplemental materials: the MLIP package user manual (https://gitlab.com/ashapeev/mlip-2-paper-supp-info/-/blob/master/manual.pdf)
[72] Novikov I S, Gubaev K, Podryabinkin E V, and Shapeev A V Supplemental material: the MLIP package usage examples (https://gitlab.com/ashapeev/mlip-2-paper-supp-info)
[73] Novikov I S, Gubaev K, Podryabinkin E V, and Shapeev A V The MLIP package (https://mlip.skoltech.ru/download/)
[74] Goreinov S A, Oseledets I V, Savostyanov D V, Tyrtyshnikov E E and Zamarashkin N L 2010 How to find a good submatrix *Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub* (Singapore: World Scientific) pp 247–56
[75] Plimpton S 1993 Fast parallel algorithms for short-range molecular dynamics *Technical Report* (Albuquerque, NM: Sandia National Labs.)
[76] Kresse G and Jürgen F 1996 Efficiency of *ab-initio* total energy calculations for metals and semiconductors using a plane-wave basis set *Comput. Mater. Sci.* **6** 15–50
[77] Kresse G and Jürgen F 1996 Efficient iterative schemes for ab *initio* total-energy calculations using a plane-wave basis set *Phys. Rev.* B **54** 11169
[78] Kresse G and Hafner J 1993 *Ab initio* molecular dynamics for liquid metals *Phys. Rev.* B **47** 558
[79] Zhu Li-F, Körmann F, Ruban A V, Neugebauer J and Grabowski B 2020 Performance of the standard exchange-correlation functionals in predicting melting properties fully from first principles: application to Al and magnetic Ni *Phys. Rev.* B **101** 144108
[80] Glass C W, Oganov A R and Hansen N 2006 USPEX—evolutionary crystal structure prediction *Comput. Phys. Commun.* **175** 713–20
[81] Ong S P, Cholia S, Jain A, Brafman M, Gunter D, Ceder G and Persson K A 2015 The materials application programming interface (API): a simple, flexible and efficient API for materials data based on representational state transfer (REST) principles *Comput. Mater. Sci.* **97** 209–15
[82] Curtarolo S *et al* 2012 Aflow: an automatic framework for high-throughput materials discovery *Comput. Mater. Sci.* **58** 218–26
[83] Rosenbrock C W, Gubaev K, Shapeev A V, Pártay L B, Bernstein N, Csányi G and Hart G L W 2019 Machine-learned interatomic potentials for alloys and alloy phase diagrams (arXiv: 1906.07816)