# PyXtal_FF: a python library for automated force field generation

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 027001 https://doi.org/10.1088/2632-2153/abc940

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
24 August 2020

REVISED
26 October 2020

ACCEPTED FOR PUBLICATION
10 November 2020

PUBLISHED
28 December 2020



TECHNICAL NOTE

# PyXtal_FF: a python library for automated force field generation

**Howard Yanxon[1], David Zagaceta[1], Binh Tang[2], David S Matteson[2] and Qiang Zhu[1]**

[1] Department of Physics and Astronomy, University of Nevada, Las Vegas, NV 89154, United States of America
[2] Department of Statistics and Data Science, Cornell University, Ithaca, NY 14853, United States of America

**E-mail:** qiang.zhu@unlv.edu

**Keywords:** machine learning potential, neural networks regression, atom-centered descriptors, atomistic simulation

## Abstract

We present PyXtal_FF—a package based on Python programming language—for developing machine learning potentials (MLPs). The aim of PyXtal_FF is to promote the application of atomistic simulations through providing several choices of atom-centered descriptors and machine learning regressions in one platform. Based on the given choice of descriptors (including the atom-centered symmetry functions, embedded atom density, SO4 bispectrum, and smooth SO3 power spectrum), PyXtal_FF can train MLPs with either generalized linear regression or neural network models, by simultaneously minimizing the errors of energy/forces/stress tensors in comparison with the data from *ab-initio* simulations. The trained MLP model from PyXtal_FF is interfaced with the Atomic Simulation Environment (ASE) package, which allows different types of light-weight simulations such as geometry optimization, molecular dynamics simulation, and physical properties prediction. Finally, we will illustrate the performance of PyXtal_FF by applying it to investigate several material systems, including the bulk $SiO_2$, high entropy alloy NbMoTaW, and elemental Pt for general purposes. Full documentation of PyXtal_FF is available at https://pyxtal-ff.readthedocs.io.

## 1. Introduction

Molecular dynamics (MD) simulations have been used routinely to model the physical behaviors of many complex systems [1–3]. The accuracy of the simulations is highly dependent on the underlying potential energy surface (PES) of the system. In principle, MD simulations can be based on *ab initio* quantum-mechanical [4] or classical force field methods. The *ab initio* MD (AIMD) simulations usually employ density functional theory (DFT) approximation [5], which can provide a reliable representation of the system. Despite the accuracy, DFT simulation can be extendable only to a few hundreds of atoms at a few picoseconds. This is due to solving the Kohn–Sham equation requires thousands of quantum-mechanical calculations that are scaled at $O(N^3)$ with respect to the number of atoms $N$. In consequence, simulating the structural evolution of many of complicated systems in DFT remains demanding in spite of the remarkable progress in computational facilities and efficient algorithms. This bottleneck in the DFT method is likely to persist in the foreseeable future. On the other hand, the classical MD method can model large systems at long-time scale, countering the unwavering issue of the DFT method. A great amount of efforts have been dedicated to developing PES using the classical method [6–10]. The reconstruction of the PES is usually based on simple analytical functions related to the scalar properties of the system. This class of force fields can be applied to gain an understanding of the qualitative behavior of the system. However, they are often inadequate in describing the quantitative properties of the system.

Recently, machine learning methods have been widely applied to resolve the dilemma in compromising between accuracy and cost [11]. The machine learning potentials (MLP) are trained by minimizing the cost function to attune the model to deliberately describe the *ab initio* data. The cost of atomistic simulation is orders of magnitude lower than the quantum mechanical simulation, allowing the system to be scaled up to $10^5$–$10^6$ atoms [12, 13].

Among many different ML models, two regression techniques are becoming increasingly popular in the materials modelling community. They include neural networks and Gaussian process regressions. The neural networks approach has an unbiased mathematical form that can adapt to any set of reference points through an iterative fitting process given 'enough' training data. The first well accepted neural networks potential (NNP) was originally applied to elemental silicon system by Behler and Parrinello [14], which demonstrated that the NNP was able to reproduce the energetic sequences of many silicon phases, as well as the radial distribution function of a silicon melt at 3000 K from DFT simulation. To gain better predictive power, they also proposed to use a series of symmetry functions (see section 2.1.1), instead of the Cartesian coordinates, as the descriptors to represent the atomic environment. Since then, many attempts have been undertaken to improve the capability of neural networks approach [15]. The accomplishments of neural networks approach have been extended to multi-component [16, 17] and organic [18] systems. In addition, Gaussian approximation potentials (GAP), in conjunction with the bispectrum coefficients of atomic neighbour density (see section 2.1.3), were first introduced to model carbon, silicon, germanium, iron, and gallium nitride [19]. GAP was further enhanced by replacing bispectrum coefficients with smooth overlapping power spectrum coefficients with explicit radial basis [20]. Similar to GAP, Thompson *et al* [21] developed (quadratic) spectral neighbor analysis potential (SNAP) method based on the Taylor expansion of bispectrum coefficients. In addition, linear regression model based on the moment tensor—comparable to atomic environments inertia tensors—as the descriptor [22] was also demonstrated to be a competitive approach. Many applications based on different MLP models have shown that machine learning potentials work remarkably well in different types of atomistic simulations [23–28].

In the recent years, several software packages [16, 29–33] were developed to train the MLPs. Among these, the RuNNer [16] is a closed source software for developing NNP, and ænet [34] is mainly written in FORTRAN/C and utilizes atom-centered symmetry functions (see section 2.1.1) as the descriptor. Similar codes, such as the n2p2 package [29] in C++, SIMPLE-NN package [30] in Python/C, and AMP package [31] in Python/FORTRAN, have similar feature as ænet package. SIMPLE-NN leverages the capability of Tensorflow platform—a deep learning GPU-accelerated library, and AMP provides several other descriptors such as Zernike and bispectrum components. Our recent works also suggested that NNP can be developed using bispectrum and power spectrum components as the descriptor while training on energy, forces, and stress simultaneously [35, 36]. Moreover, DeepPot-SE [37] and SchNetPack [33] packages introduce additional filters to the descriptor such as distance-chemical-species-dependent filter and continuous convolutional filter, respectively, prior to the deep learning model.

In this paper, we present PyXtal_FF—an open-source package in Python scripting language—for developing MLPs such as NNP and generalized linear potential (GLP). The objective of PyXtal_FF is to provide handy user-interface in developing MLPs with training of energy, force, and stress contributions simultaneously. PyXtal_FF creates MLPs based on atom-centered descriptors such as (weighted) atom-centered symmetry functions [11], embedded atom density [38], SO4 bispectrum coefficients [19], and smooth SO3 power spectrum [20]. Finally, we will demonstrate the usage of the current features of the package with $SiO_2$ [30], high entropy alloy [39], and elemental Pt [37] as examples.

## 2. Theory

In this section, we will provide in-depth discussions of the two main ingredients in creating MLP: the atom-centered descriptor and regression technique. The construction of the total energy of a crystal structure can be written as the collections of atomic energy contributions, in which is a functional ($E$) of the atom-centered descriptor ($\boldsymbol{X}_i$):

$$E_{\text{total}} = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N} E_i(\boldsymbol{X}_i). \tag{1}$$

Specifically, the functional represents regression techniques such as neural networks or generalized linear regressions.

Since neural networks and generalized linear regressions have well-defined functional forms, the analytic derivatives can be derived by applying the chain rule to obtain the force at each atomic coordinate, $\boldsymbol{r}_m$:

$$\boldsymbol{F}_m = -\sum_{i=1}^{N} \frac{\partial E_i(\boldsymbol{X}_i)}{\partial \boldsymbol{X}_i} \cdot \frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{r}_m}. \tag{2}$$

```python
import numpy as np

"""
Einstein summation to compute force and stress.
dedx: 2D array [N, L]
dxdr: 4D array [N, N, L, 3]
rdxdr: 5D array [N, N, L, 3, 3]
force: 2D array [N, 3]
stress: 2D array [3, 3]
"""
force = -np.einsum("ik, ijkl->jl", dedx, dxdr)
stress = -np.einsum("ik, ijklm->lm", dedx, rdxdr)
```

**Listing 1.** Force and stress computation in Python.

Force is an important property to accurately describe the local atomic environment especially in geometry optimization and MD simulation. Finally, the stress tensor is acquired through the virial stress relation:

$$\boldsymbol{S} = -\sum_{m=1}^{N} \boldsymbol{r}_m \otimes \sum_{i=1}^{N} \frac{\partial E_i(\boldsymbol{X}_i)}{\partial \boldsymbol{X}_i} \cdot \frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{r}_m}, \tag{3}$$

where $\otimes$ is the outer product.

According to equations (2) and (3), one needs to compute the energy derivative $\frac{\partial E}{\partial \boldsymbol{X}}$ and the derivatives of descriptor $X$ with respect to the atomic positions. For a structure with $N$ atoms and $L$ descriptors, the energy derivative is a 2D array of $[N, L]$. The force related derivative (dxdr) can be best organized as a 4D array with the dimension of $[N, N, L, 3]$. Note that dxdr$[i, j, :, :]$ is zero when the *i-j* atomic pair has a distance larger than the cutoff distance. Thus, it may become a sparse array when the structure has a large number of atoms. Correspondingly, one can easily derive the 5D rdxdr array by multiplying $r$ to each dxdr according to the outer product. In Python, one can simply compute the forces and stresses based on the following Einstein summation.

### 2.1. Atom-centered descriptors

Descriptor—a representation of a crystal structure—plays a critical role in constructing reliable MLP. If the MLP is directly mapped from the atomic positions or the Cartesian coordinates, it can only describe systems with the same number of atoms due to the fixed length of the regression input. In addition, Cartesian coordinates are poor descriptors in describing the structural environment of the system, restricted by the periodic boundary conditions. While the total energy of the structure remains the same by translation, rotational, or permutation operations, the atomic positions will change. Several types of descriptors have been developed in the past few years [40]. For example, Coulomb matrix has been widely used due to its simplicity. Coulomb matrix encompasses self interaction based on the nuclear charge and Coulomb repulsion between two nuclei [41, 42]. Logically, the Coulomb matrix can be upgraded for periodic crystals through Ewald summation that includes long range interaction calculated in reciprocal space. In addition, many-body tensor representation—derives from Coulomb matrix while related to bag of bonds which corresponds to different types of bonding in molecular systems—can be used for both finite and periodic systems when interpretability/visualization is desirable [43]. These descriptors have been widely used to model the molecules.

In the atom-centered descriptors, one usually needs to consider the neighboring environment for the centered atom within a cutoff radius of $R_c$. To ensure the descriptor mapping from the atomic positions smoothly approaching zero at the $R_c$, a cosine cutoff function $(f_c)$ is included to every mapping scheme:

$$f_c(R) = \begin{cases} \frac{1}{2}\cos\left(\pi \frac{R}{R_c}\right) + \frac{1}{2} & R \le R_c, \\ 0 & R > R_c \end{cases}, \tag{4}$$

where $R$ is distance. The cutoff function is zero at $R_c$ and the intensity decreases as $R$ approaches $R_c$. Consequently, the derivative of the cosine cutoff function is

$$\frac{\partial f_c}{\partial R} = \begin{cases} -\frac{\pi}{2R_c}\sin\left(\pi \frac{R}{R_c}\right) & R \le R_c, \\ 0 & R > R_c \end{cases}. \tag{5}$$

One should heed the importance of the vanishing derivative of the cutoff function at $R_c$, which is important in describing the force. By definition, there is no discontinuity as the slope decays to zero at $R_c$. Additionally, other cutoff functions are available in PyXtal_FF: hyperbolic tangent, exponential and polynomials functions [44].

In the following, we will introduce four types of atom-centered descriptors in details. The corresponding derivative terms can be found in A, B and our recent work [36].

*2.1.1. (Weighted) atom-centered symmetry functions (G)*

The atom-centered symmetry functions (ACSFs) are the very first types of descriptors used in the MLP development [14]. In general, there are two classes of ACSFs: radial and angular symmetry functions [11]. The radial symmetry function or $G^{(2)}$ describes the radial distribution of the atomic environment, and the angular symmetry functions, $G^{(4)}$ and $G^{(5)}$, account for the three-body angular distribution of atoms in the neighborhood. The $G^{(2)}$ is expressed as the sum of the radial distances between the center atom $i$ and the neighbor atoms $j$ as follows:

$$G_i^{(2)} = \sum_{j \neq i} e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}). \tag{6}$$

Here, $G^{(2)}$ value is controlled by the width ($\eta$) and the shift ($R_s$).

$G^{(4)}$ and $G^{(5)}$ symmetry functions are a few of many ways to capture the angular information via three-body interactions ($\theta_{ijk}$). As the structures are constraint by the periodic boundary condition, a three-body periodic description such as $\cos(\theta_{ijk})$ is used. The explicit form of $G^{(4)}$ and $G^{(5)}$ are

$$G_i^{(4)} = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} [(1 + \lambda \cos\theta_{ijk})^{\zeta} \cdot e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \cdot f_c(R_{jk})] \tag{7}$$

$$G_i^{(5)} = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} [(1 + \lambda \cos\theta_{ijk})^{\zeta} \cdot e^{-\eta(R_{ij}^2+R_{ik}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik})] \tag{8}$$

$\zeta$ determines the strength of angular information. The degree of $\zeta$ is normalized by $2^{1-\zeta}$ for unvarying the values of $G^{(4)}$ and $G^{(5)}$ symmetry functions due to ranges of $\zeta$. $\lambda$ values are set to $+1$ and $-1$, for inverting the shape of the cosine function. The difference between $G^{(4)}$ and $G^{(5)}$ symmetry functions is in the interactions between the neighbors $j$ and $k$. The modification in $G^{(5)}$ symmetry function yields in dampening value of $G^{(5)}$, which can be beneficial in representing larger atomic separation between the two neighbors.

Clearly, the number of ACSFs will grow depending on chemical species as the separations of chemical species are needed. For instance, in a binary AB system, the number of $G^{(2)}$ ACSFs on specie A need to double to distinguish A–A and A–B pair interactions. For $G^{(4)}$, three different triplets A–A–A, A–A–B, B–A–B (where the middle position denotes the center atom) will be needed. To avoid this unpleasant growth, one can apply a weighting parameter based on the chemical species when counting these atomic pairs and triplets. One popular choice is simply to use the atomic number as the weighting parameters. Hence, Gastegger and coauthors proposed the weighted version of ACSF [45], in which each component of the radial and angular symmetry functions in equations (6)–(8) can be multiplied by the following:

$$\text{the weighted ACSF:} \quad \begin{cases} Z_j & \text{radial,} \\ Z_j Z_k & \text{angular} \end{cases}$$

where $Z_j, Z_k$ represents the atomic number of neighboring atom $j$ and $k$.

To obtain a satisfactory MLP model, one has to choose a set of parameters to construct the (w)ACSF descriptors, which may require demanding human intervention [33, 37, 45]. As mentioned, the choice of $\lambda$ is straightforward. In general, $\zeta$ takes the value of 1. Increasing $\zeta$ focuses on the strength of the angular information in region close to $0^\circ$ and $180^\circ$, and decreasing it will weaken the contribution of angular information at around $90^\circ$. Since the exponential term has larger effect on the symmetry functions, the selection of $\eta$ and $R_s$ can be more elaborate. Several routines are available in the literature [38, 45, 46] by fixing $\eta$ while varying $R_s$ or vice versa.

*2.1.2. Embedded atom density ($\rho$)*

Embedded atom density (EAD) descriptor [38] is inspired by embedded atom method (EAM)—description of atomic bonding by assuming each atom is embedded in the uniform electron cloud of the neighboring atoms [6, 47]. In EAD, the electron density is modified by including the square of the linear combination the atomic orbital components:

$$\rho_i = \sum_{l_x,l_y,l_z}^{l_x+l_y+l_z=L_{\max}} \frac{L_{\max}!}{l_x!l_y!l_z!}\left(\sum_{j\neq i}^{N} Z_j \Phi(R_{ij})\right)^2 \tag{9}$$

where $Z_j$ represents the atomic number of neighbor atom $j$. $L_{\max}$ is the quantized angular momentum, and $l_{x,y,z}$ are the quantized directional-dependent angular momentum. For example, $L_{\max} = 2$ corresponds to the $d$ orbital. Lastly, the explicit form of $\Phi$ is

$$\Phi(R_{ij}) = \frac{x_{ij}^{l_x} y_{ij}^{l_y} z_{ij}^{l_z}}{R_c^{l_x+l_y+l_z}} \cdot e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}). \tag{10}$$

According to quantum mechanics, $\rho$ follows the similar procedure in determining the probability density of the states.

EAD can be regarded as an alternative version of ACSF without classification between the radial and angular term. The angular or three-body term is implicitly incorporated in when $L_{\max} > 0$ [38]. By definition, the computation cost for calculating EAD is cheaper than angular symmetry functions by avoiding the extra sum of the $k$ neighbors. In term of usage, the parameters $\eta$ and $R_s$ are similar to the strategy used in the Gaussian symmetry functions, and the maximum value for $L_{\max}$ is 3, i.e. up to $f$ orbital.

*2.1.3. SO(4) bispectrum (B)*

The SO(4) bispectrum components [19–21] are another type of atom-centered descriptor based on the harmonic analysis of the atomic neighbor density function on the 3-sphere. The atomic neighbor density function is given by [20]

$$\rho(\boldsymbol{r}) = \delta(\boldsymbol{r}) + \sum_{i}^{R_c} w_i f_c(\boldsymbol{r}_i)\delta(\boldsymbol{r}-\boldsymbol{r_i}) \tag{11}$$

where $w_i$ is a species dependent weight factor and $f_c$ is a cutoff function. The cutoff function is $f_c$ is introduced to ensure that the atomic neighbor density function goes smoothly to zero at the cutoff.

Then we map the atomic neighbor density function from 3D euclidean space to another 3D space, the surface of a four dimensional hypersphere:

$$\begin{aligned} s_1 &= r_0 \cos\omega \\ s_2 &= r_0 \sin\omega \cos\theta \\ s_3 &= r_0 \sin\omega \sin\theta \cos\phi \\ s_4 &= r_0 \sin\omega \sin\theta \sin\phi, \end{aligned}$$

where $r_0$ is a parameter and the polar angles are defined by

$$\begin{aligned} \theta &= \arccos\left(\frac{z}{r}\right) \\ \phi &= \arctan\left(\frac{y}{x}\right) \\ \omega &= \frac{\pi r}{r_0}. \end{aligned} \tag{12}$$

The Winger-D matrix elements ($D^j_{m',m}$) are the harmonic functions on the 3-sphere, therefore an arbitrary function defined on the 3-sphere can be expanded in terms of Wigner-D matrix elements. Here we expand the atomic neighbor density function on the 3-sphere in terms of Wigner-D matrices:

$$\rho(\boldsymbol{r}) = \sum_{j=0}^{+\infty} \sum_{m',m=-j}^{+j} c^j_{m',m} D^j_{m',m}(\omega;\theta,\phi),$$

where the expansion coefficients $c^{j}_{m',m}$ are given by the following inner product:

$$c^{j}_{m',m} = \left\langle D^{j}_{m',m} | \rho(\boldsymbol{r}) \right\rangle = D^{*j}_{m',m}(\mathbf{0}) + \sum_{i}^{r_i \leq R_c} f_c(r_i) D^{*j}_{m',m}(\omega_i; \theta_i, \phi_i). \tag{13}$$

Finally, the SO(4) bispectrum components can then be calculated using third order products of the expansion coefficients:

$$B^{j_1,j_2,j}_{i} = \sum_{m',m=-j}^{j} c^{*j}_{m',m} \sum_{m'_1,m_1=-j_1}^{j_1} c^{j_1}_{m'_1,m_1} \times \sum_{m'_2,m_2=-j_2}^{j_2} c^{j_2}_{m'_2,m_2} C^{jj_1j_2}_{mm_1m_2} C^{jj_1j_2}_{m'm'_1m'_2}, \tag{14}$$

where $C$ is a Clebsch–Gordan coefficient.

*2.1.4. Smooth SO(3) power spectrum (P)*

The Smooth SO(3) Power Spectrum components were been proposed to describe the atomic local environment [20]. In contrast to the SO(4) bispectrum components, the Smooth SO(3) power spectrum is based on an alternative atomic neighbor density while also expanded on the 2-sphere and a radial basis. The alternative atomic neighbor density is defined in terms of Gaussians as follows:

$$\rho'(\boldsymbol{r}) = \sum_{i}^{r_i \leq R_c} w_i e^{-\alpha |\boldsymbol{r} - \boldsymbol{r_i}|^2}. \tag{15}$$

Then the atomic neighbor density function is then expanded in terms of spherical harmonics and a radial basis $g_n(r)$ as shown in equation (15):

$$\rho'(\boldsymbol{r}) = \sum_{l=0}^{+\infty} \sum_{m=-'l}^{+l} c_{nlm} g_n(r) Y_{lm}(\hat{\boldsymbol{r}}),$$

where the expansion coefficients $c_{nlm}$ are given by

$$\begin{aligned} c_{nlm} &= \langle Y_{lm} g_n(r) | \rho' \rangle \\ &= 4\pi \sum_{i}^{r_i \leq R_c} w_i e^{-\alpha r_i^2} Y^*_{lm}(\hat{\boldsymbol{r}}_{\boldsymbol{i}}) \times \\ &\quad \int_0^{R_c} r^2 g_n(r) I_l(2\alpha r r_i) e^{-\alpha r^2} dr, \end{aligned}$$

where $I_l$ is a modified spherical Bessel function of the first kind. A convenient radial basis for this purpose, $g_n(r)$, consisting of cubic and higher order polynomials, orthonormalized on the interval $(0, R_c)$ has been suggested by Bartok [20]:

$$g_n(r) = \sum_{\alpha} W_{n,\alpha} \phi_\alpha(r), \tag{16}$$

where $W_{n,\alpha}$ are the orthonormalization coefficients given by the relation to the overlap matrix $\boldsymbol{S}$ by $\boldsymbol{W} = \boldsymbol{S}^{-1/2}$ and

$$\phi_\alpha(r) = (R_c - r)^{\alpha+2} / N_\alpha$$

$$N_\alpha = \sqrt{\frac{2 r_{\text{cut}}^{(2\alpha+7)}}{(2\alpha+5)(2\alpha+6)(2\alpha+7)}}.$$

The elements of the overlap matrix $\boldsymbol{S}$ are given by

$$\begin{aligned} S_{\alpha\beta} &= \int_0^{r_{\text{cut}}} r^2 \phi_\alpha(r)\phi_\beta(r)dr \\ &= \frac{\sqrt{(2\alpha+5)(2\alpha+6)(2\alpha+7)(2\beta+5)(2\beta+6)(2\beta+7)}}{(5+\alpha+\beta)(6+\alpha+\beta)(7+\alpha+\beta)} \end{aligned} \tag{17}$$

and finally, the Smooth SO(3) power spectrum is given by

$$P_i^{n_1 n_2 l} = \sum_{m=-l}^{+l} c_{n_1 lm} c^*_{n_2 lm}. \tag{18}$$

### 2.2. Regression models

Here, we discuss the regression model, i.e. the functional form ($E$) presented in equation (1). Each regression model is species-dependent, i.e. as the the number of species increases, the regression parameters will increase. For the sake of simplicity, we will explain the regression models for the single-species system.

In any regression model, the objective is to minimize a loss function which describes the discrepancies between the prediction and true reference values (including energy, force, and stress tensors) for each atomic configuration in the training dataset:

$$\begin{aligned} \Delta = &\frac{1}{2\,M}\sum_{i=1}^{M}\left[\left(\frac{E_i - E_i^{\text{Ref}}}{N_{\text{atom}}^i}\right)^2 + \frac{\beta_f}{3N_{\text{atom}}^i}\sum_{j=1}^{3N_{\text{atom}}^i}(F_{i,j} - F_{i,j}^{\text{Ref}})^2 \right. \\ &\left. + \frac{\beta_s}{6}\sum_{p=0}^{2}\sum_{q=0}^{p}(S_{pq} - S_{pq}^{\text{Ref}})^2\right], \end{aligned} \tag{19}$$

where $M$ is the total number of structures in the training pool, and $N_i^{\text{atom}}$ is the total number of atoms in the $i$th structure. The superscript Ref corresponds to the target property. $\beta_f$ and $\beta_s$ are the force and stress coefficients respectively. They scale the importance between energy, force, and stress contribution as the force and stress information can overwhelm the energy information due to their sizes. Additionally, a regularization term can be added to induce penalty on the entire parameters preventing overfitting:

$$\Delta_{\text{p}} = \frac{\alpha}{2\,\text{M}}\sum_{i=1}^{m}(\boldsymbol{w}^i)^2, \tag{20}$$

where $\alpha$ is a dimensionless number that controls the degree of regularization.

Clearly, one has to choose differentiable functional as well as its derivative due to the existence of force ($F$) and stress ($S$) contribution along with the energy ($E$) in the loss function. In the following sections, generalized linear regression and neural network regression will be introduced.

#### *2.2.1. Generalized linear regression*

This regression methodology is a type of polynomial regression. Essentially, the quantum-mechanical energy, forces, and stress can be expanded via Taylor series with atom-centered descriptors as the independent variables:

$$E_{\text{total}} = \gamma_0 + \boldsymbol{\gamma}\cdot\sum_{i=1}^{N_{\text{atom}}}\boldsymbol{X}_i + \frac{1}{2}\sum_{i=1}^{N_{\text{atom}}}\boldsymbol{X}_i^T\cdot\boldsymbol{\Gamma}\cdot\boldsymbol{X}_i + \cdots, \tag{21}$$

where $N_{\text{atom}}$ is the number of atoms in a structure. $\gamma_0$ and $\boldsymbol{\gamma}$ are the weights presented in scalar and vector forms. $\boldsymbol{\Gamma}$ is the symmetric weight matrix (i.e. $\boldsymbol{\Gamma}_{12} = \boldsymbol{\Gamma}_{21}$) describing the quadratic terms. If each $\boldsymbol{X}_i$ has a length of $d$, the length of $\boldsymbol{\gamma}$ is $d$ and $\boldsymbol{\Gamma}$ is a $d \times d$ matrix. In this equation, we only restricted the expansion up to polynomial 2 due to to enormous increase in the weight parameters.

In consequence, the force on atom $j$ and the structural stress matrix can be derived according to equations (2) and (3), respectively:

$$\boldsymbol{F}_m = -\sum_{i=1}^{N_{\text{atom}}}\left(\boldsymbol{\gamma}\cdot\frac{\partial\boldsymbol{X}_i}{\partial\boldsymbol{r}_m} + \frac{1}{2}\left[\frac{\partial\boldsymbol{X}_i^T}{\partial\boldsymbol{r}_m}\cdot\boldsymbol{\Gamma}\cdot\boldsymbol{X}_i + \boldsymbol{X}_i^T\cdot\boldsymbol{\Gamma}\cdot\frac{\partial\boldsymbol{X}_i}{\partial\boldsymbol{r}_m}\right]\right) \tag{22}$$

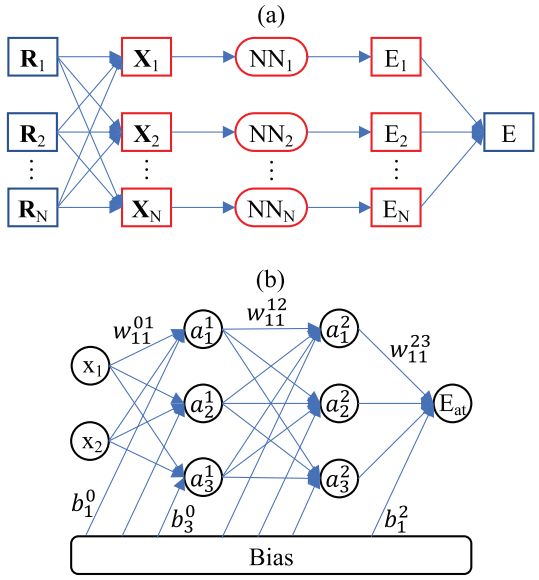


**Figure 1.** (a) A schematic diagram of high-dimensional neural networks. (b) A zoom-in version of the color-coded part in (a).

```
from pyxtal_ff import PyXtal_FF

# define the path of train/test data
train = 'train.json'
test = 'test.json'

# define the descriptor
descriptor = {'type': 'Bispectrum',
              'parameters': {'lmax': 3},
              'Rc': 4.9}

# define the regression model
model = {'system' : ['Pt'],
         'hiddenlayers': [16, 16],
         'epoch': 1000,
         'path': 'Pt-Bispectrum/'
         'optimizer': {'method': 'lbfgs'}}

# define the pyxtal_FF model and train it
mlp = PyXtal_FF(descriptor, model)
mlp.run(TrainData=train, TestData=test)
```

**Listing 2.** PyXtal_FF script for force field training.

$$\boldsymbol{S} = -\sum_{m=1}^{N_{\text{atom}}} \boldsymbol{r}_m \otimes \sum_{i=1}^{N_{\text{atom}}} \left( \gamma \cdot \frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{r}_m} + \frac{1}{2} \left[ \frac{\partial \boldsymbol{X}_i^T}{\partial \boldsymbol{r}_m} \cdot \boldsymbol{\Gamma} \cdot \boldsymbol{X}_i + \boldsymbol{X}_i^T \cdot \boldsymbol{\Gamma} \cdot \frac{\partial \boldsymbol{X}_i}{\partial \boldsymbol{r}_m} \right] \right). \tag{23}$$

Note that the energy, force, and stress share the weight parameters $\{\gamma_0, \gamma_1, ..., \gamma_n, \boldsymbol{\Gamma}_{11}, \boldsymbol{\Gamma}_{12}, ..., \boldsymbol{\Gamma}_{nn}\}$, where $n$ is total the number of descriptors of the center atom. Once the energy, force and stress tensors are known, the derivative of the loss function can be evaluated. Finding the zero derivative of loss function (equation (19)) in linear regression is equivalent to solve a set of linear equations of $\boldsymbol{Ax} = \boldsymbol{b}$. In PyXtal_FF, we construct such $\boldsymbol{A}$ matrix and use the *numpy.linalg.lstsq* solver to obtain the least-squares solution.

### *2.2.2. Neural network regression*

Compared to the linear regression, neural networks provide more flexible functionals to fit a large datasets. Figure 1 shows a schematic diagram based on neural networks training. Prior to the neural networks architecture, the atom-centered descriptors are mapped based on the atomic environment of a structural configuration as discussed in the previous section. These descriptors serve as the input to the neural

```
from pyxtal_ff import PyXtal_FF
from pyxtal_ff.calculator import PyXtalFFCalculator
    , optimize
from ase.io import read

# load the trained model
mliap = "Pt-Bispectrum/16-16-checkpoint.pth"
ff = PyXtal_FF(model={'system': ["Pt"]},
               logo=False)
ff.run(mode='predict', mliap=mliap)
calc = PyXtalFFCalculator(calc)

# read the structure
pt_bulk = read('Pt_bulk.cif')

# perform the relaxation
pt_bulk.set_calculator(calc, box=True)
pt_bulk = optimize(pt_bulk)
print('energy: ', pt_bulk.get_potential_energy())
```

**Listing 3.** PyXtal_FF script to perform geometry optimization.

networks architecture and are arranged in the first layer as shown in figure 1(b). The next layers are the hidden layers. Neural networks can simply cast more weights parameters as needed through increasing number of hidden layers and/or hidden layers nodes without the increasing number of descriptors. The nodes in the hidden layers carry no physical meaning.

For each of the hidden nodes, activation functions such as Tanh and Sigmoid functions are frequently used in our NNP implementation. While ReLU as an activation function is extremely popular in image processing, we believe ReLU is not an appropriate choice in constructing MLP, due to the function carries discontinuity at zero. These nodes are connected via the weights and biases and propagate in forward direction only. In the end, the output node represents the atomic energy. A mathematical form to determine any node value can be written as

$$X^l_{n_i} = a^l_{n_i}\left(b^{l-1}_{n_i} + \sum_{n_j=1}^{N} W^{l-1,l}_{n_j,n_i} \cdot X^{l-1}_{n_j}\right). \tag{24}$$

The value of a neuron ($X^l_{n_i}$) at layer $l$ can determined by the relationships between the weights ($W^{l-1,l}_{n_j,n_i}$), the bias ($b^{l-1}_{n_i}$), and all neurons from the previous layer ($X^{l-1}_{n_j}$). $W^{l-1,l}_{n_j,n_i}$ specifies the connectivity of neuron $n_j$ at layer $l-1$ to the neuron $n_i$ at layer $l$. $b^{l-1}_{n_i}$ represents the bias of the previous layer that belongs to the neuron $n_i$. These connectivity are summed based on the total number of neurons ($N$) at layer $l-1$. Finally, an activation function ($a^l_{n_i}$) is applied to the summation to induce non-linearity to the neuron ($X^l_{n_i}$). $X_{n_i}$ at the output layer is equivalent to an atomic energy, and it represents an atom-centered descriptor at the input layer. The collection of atomic energy contributions are summed to obtain the total energy of the structure. At the end, the total energy, forces and stress tensors are compared to the reference values (see equation (19)). This process is called forward propagation.

Similar to the linear regression, one needs to obtain a set of weight parameters to minimize the loss function. In NN architecture, the gradient of loss with respect to the weight parameters can be conveniently done by the backpropagation algorithm. Hence, a number of optimization algorithms can be applied here to update the weights iteratively, until the optimal solution is found.

## 3. PyXtal_FF workflow

In this section, we discuss about development of PyXtal_FF and its philosophy. PyXtal_FF is written in Python. Presently, the package is equipped with two regression models and four types atom-centered descriptor, as explained in section 2. These regression models and atom-centered descriptors are easily extendable without changing the core user-interface features. Figure 2 represents the workflow of PyXtal_FF.

First, PyXtal_FF utilizes the Atomic Simulation Environment (ASE) package [48] to parse the DFT data assembled in several formats, including ASE database, JSON, extended XYZ and the VASP OUTCAR formats. Further, ASE is employed to compile the atomic neighborhood of each atom in the unit cell based on the periodic boundary conditions within a cutoff radius. After the neighboring data are gathered, it will compute the user-defined type of descriptor. The computation follows the theory described in section 2.1

```
# ACSF (70)
para = {'G2':
        {'eta': [0.003214, 0.035711,
                 0.071421, 0.124987,
                 0.214264, 0.357106,
                 0.714213, 1.428426],
         'Rs': [0]},
        'G4':
        {'lambda': [-1, 1],
          'zeta':[1, 2, 4],
          'eta': [0.000357, 0.028569, 0.089277]}
        }
descriptor = {'type': 'ACSF',
              'Rc': 4.9,
              'parameters': para,
             }

# wACSF (26)
descriptor = {'type': 'wACSF',
              'Rc': 4.9,
              'parameters': para,
             }

# EAD (30)
para = {'eta': [0.003214, 0.035711,
                0.071421, 0.124987, 0.214264],
        'Rs': [0, 1.50],
        'lmax': 2,
       }

descriptor = {'type': 'EAD',
              'Rc': 4.9,
              'parameters': para,
             }

# SO3 (40)
descriptor = {'type': 'SO3',
              'Rc': 4.9,
              'parameters': {'nmax': 4,
                             'lmax': 3},
             }

# SO4 (30)
descriptor = {'type': 'SO4',
              'Rc': 4.9,
              'parameters': {'lmax': 3},
             }
```

**Listing 4.** PyXtal_FF script to define the descriptors.

utilizing NumPy—a Python library for scientific computing [49]. For every structure, the descriptor calculator will return the descriptors and the force and stress related derivatives. Eventually, the descriptors represent the independent variables in the regression models to obtain the energy, and the derivative terms are needed to compute the force and stress values.

For the regression, the Pyxtal_FF supports two models, linear (quadratic) regression and neural networks. Here we focus on the latter since it is a more popular workforce for MLP development. The neural network regression is powered by PyTorch [50]—an open-source deep learning framework based on automatic differentiation [51]. Currently, we support three optimization algorithms for training: Limited Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [52], adaptive Moment Estimation (Adam) [53], and stochastic gradient descent (SGD) with momentum [54]. The L-BFGS method with approximated line search is the recommended optimizer when the training data is relatively small, as the quasi-Newton method is generally more stable and finds local optima more efficiently. With larger training datasets, however, the L-BFGS method is memory demanding, and one can seek to use first-order methods such as Adam or SGD with momentum. Both SGD and Adam algorithms are usually done in mini-batches, where the gradients for each weight update are calculated based on a subset of the entire training dataset. Training in mini batches can reduce the variance of the parameter updates leading to stable convergence. If needed, the training can also be done in graphical processing units (GPU) mode.

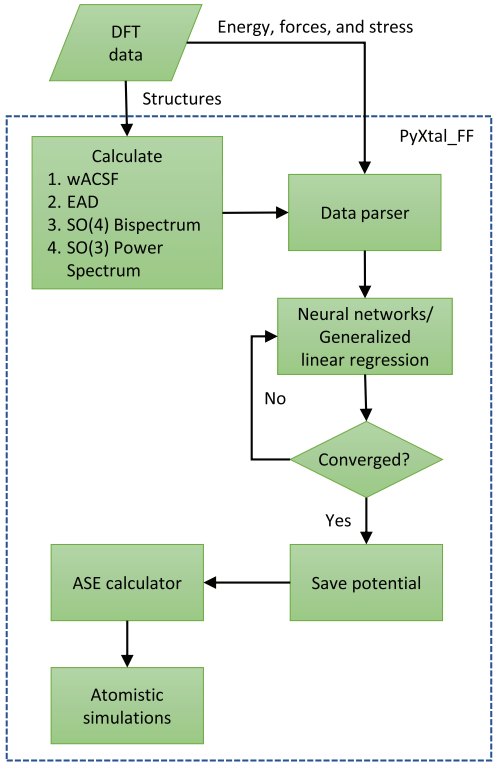


**Figure 2.** Schematic diagram of PyXtal_FF workflow for NNP/GLP training. In the neural network, data parser includes normalization of the calculated descriptors.

In addition to the force field generation, PyXtal_FF also provides the supports to utilize the trained models for several types of atomistic simulations, including geometry optimization, MD simulation, physical properties prediction, and phonon calculation. These features are managed by ASE calculator, in which the MLP potential passes the energy, forces, and stress tensors to the calculator and ASE performs the relevant atomistic simulations. Since these simulation will be powered by Python, we only recommend to use them for light weight simulations. In the near future, we are going to work on interfacing the trained MLP with LAMMPS [55] to enable the large-scale atomistic simulation.

## 4. Example usage

PyXtal_FF can be used as stand-alone library in Python scripts. A PyXtal_FF example code to train Pt model is shown in the following listing

The atom-centered descriptors and the model are described in dictionary. The dictionary keys determine the necessary command for the code and are made as intuitive as possible. Most of keys follow the hyperparameters in the section 2. By default, PyXtal_FF will use neural networks as the regression algorithm. Here, PyXtal_FF will look for *train.json* and *test.json* files as the training and test datasets, respectively.

After the training is complete, the trained model is saved in the result folder (*Pt-Bispectrum*) with a name of *16-16-checkpoint.pth*, in which 16-16 denotes two hidden layers with 16 nodes each. PyXtal_FF provides a built-in interface with the ASE code, in which one can use the model to perform different types of calculations through ASE. Below is a simple example to perform the geometry optimization on a Pt bulk crystal (*Pt_bulk.cif*) based on the trained model from the listing 2.

In addition to geometry optimization and MD simulation, PyXtal_FF also provides several utility functions to simulate the elastic and phonon properties, which are based on several external Python libraries including Phonopy [56], seekpath [57] and matscipy [58]. More detailed examples can be found in the online documentation https://pyxtal-ff.readthedocs.io.

## 5. Applications

In this section, we choose three different examples to illustrate the power of PyXtal_FF and benchmark the performances of different descriptors. While the linear regression scheme is also supporte by PyXtal_FF, we will focus on the NNP model as it provides more flexibility. The examples to be investigated mainly differ by

**Table 1.** The MAE values of the predicted energy and forces of 1316 $SiO_2$ dataset from the 30-30 neural network models with different descriptors within 12 000 L-BFGS steps of training. For each type of descriptors, the average CPU time for descriptor computation per structure is also given.

| | CPU time (s/60 atoms) | Energy (meV atom$^{-1}$) | Force (meV Å$^{-1}$) |
|---|---|---|---|
| ACSF (70) | 4.374 | 1.3 | 81.2 |
| wACSF (26) | 4.372 | 2.1 | 141.8 |
| EAD (30) | 0.584 | 4.8 | 259.0 |
| SO3 (40) | 1.028 | 1.4 | 115.1 |
| SO4 (30) | 1.078 | 3.3 | 204.2 |

the source of datasets, including (1) single $SiO_2$ from pure MD simulation; (2) collective dataset of NbMoTaW from various approaches; (3) elemental Pt consisting of bulk, surfaces and clusters from different runs of MD simulations.

### 5.1. Binary system

The $SiO_2$ dataset [30] was generated by the DFT method within the framework of VASP [59], using the generalized gradient approximation Perdew–Burke–Ernzerhof (PBE) exchange-correlation functional [60]. The kinetic energy cutoff was set to 500 eV, and the energy convergence criterion for constructing the **k**-point mesh is within 10 meV/atom. The MD trajectories are taken at different temperatures including liquid, amorphous and crystalline ($\alpha$-quartz, $\alpha$-cristobalite, and tridymite) configurations. The original dataset contain 3,048 $SiO_2$ configurations (60 atoms per structure). For simplicity, we considered a subset that consists of 1,316 structures. with the goal of to gaining an overview of performances and computation costs for each descriptor. Below gives the parameters to define each descriptor.

In short, we choose a universal cutoff value of 4.9 Å for all descriptors. Each descriptors requires some manual selection of hyperparameters in the real (e.g. $\eta, \lambda, \zeta, R_s$) or integer ($l_{\max}$, $n_{\max}$) space. The ACSF parameters were taken from reference [30] which lead to 70 descriptors. In its wACSF version, the number is reduced to 26. For EAD, we chose a similar set of parameters for $\eta$ and $R_s$, which make 30 descriptors when $L_{\max} = 2$. For SO3 and SO4, only the integer type hypeparameters need to be provided. In this work, we set 40 SO3 descriptors with $n_{\max} = 4$ and $l_{\max} = 3$, and 30 SO4 descriptors with $l_{\max} = 4$. The neural network regression will be used with two hidden layers with 30 nodes each.

Table 1 summarizes the performances of each training after 12 000 steps. First, the ACSF-70 set yields the best accuracy in both energy (1.3 meV atom$^{-1}$) and forces (81.2 meV Å$^{-1}$), while the errors in its corresponding wACSF-26 set rise by 60%–70% in both energy (2.1 meV atom$^{-1}$) and forces (141.8 meV Å$^{-1}$). On the other hand, the weighted EAD-30 descriptor, supposed to mimic G2 and G4 ACSFs, gives the highest errors (4.0 meV atom$^{-1}$ for energy and 300 meV Å$^{-1}$ for forces). This may be due to lack of optimization on the hyperparameters. However, it should be noted that the computation of EAD is much faster than ACSF. Therefore, it is worth exploring a systematic approach to obtain the optimum set for EAD. For the two spectral descriptors, SO3-40 seems to outperform SO4-30 while it cost about a similar level of CPU time. In terms of accuracy, SO3-40 (1.4 meV atom$^{-1}$ in energy MAE and 115.1 meV Å$^{-1}$ in force MAE) is in the middle of ACSF-70 and wACSF-26. Another remarkable advantage of the spectral descriptors is that tuning the hyperparameters is much easier. If one does not want to spend too much time on choosing the hyperparameters, SO3 seems to be a better choice than ACSF. We note that all descriptor computations are based on Python. It is expected that the speed will be much faster when they are implemented in FORTRAN or C languages.

### 5.2. High entropy alloy

High entropy alloys (HEAs) are systems that encompass four or more equimolar/near-equimolar alloying elements. It has been shown that HEAs carry many interesting properties such as high hardness and corrosion resistance [61, 62]. Due to the high computational cost of DFT method, HEA serves as a great example in MLP development with PyXtal_FF. Here, we will use NbMoTaW HEA as an example [39], which are comprised of elemental, binary, ternary, and quaternary systems. Each of the elemental systems has their ground state, strain-distorted, surface, and AIMD configurations. The binary alloys are composed of solid solution structures with the size of $2 \times 2 \times 2$ supercell. Lastly, 300 K, 1000 K, and 3000 K AIMD configurations along with special quasi-random structures establish the ternary and quaternary data points.

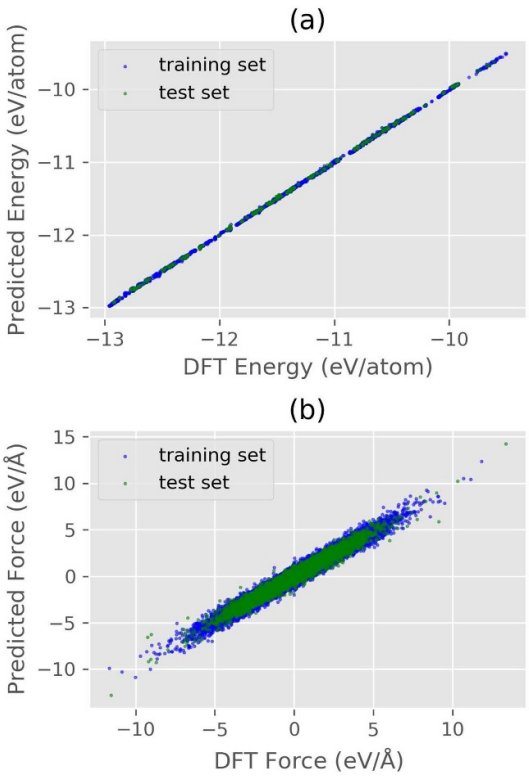


**Figure 3.** The correlation plots between NNP and DFT for (a) energy and (b) forces in the HEA system. The energy MAE values are 6.69 and 7.57 meV atom$^{-1}$ for training and test sets, while the force MAE values are 0.14 and 0.17 eV Å$^{-1}$.

The total structures used in developing the MLP are 5529 configurations for training set and 376 configurations for test set.

In the original work [39], SNAP based on the linear regression predicted that the MAE values for energies are 4.3 and 5.1 meV atom$^{-1}$ for the training and test sets, and the MAE values in forces are 0.13 and 0.14 eV Å$^{-1}$. The results demonstrated a quite satisfactory accuracy comparable to the quantum calculation. However, it needs to be noted that the energy training in reference [39] was based on the comparison of formation energy relative to the elemental solids, which spans from −0.193 to 0.934 eV atom$^{-1}$ for the entire dataset, whereas the atomic energy spans from −12.960 to −9.502 eV atom$^{-1}$. Training with the energy in a normalized range can surely reduce the error of fitting. However, this fitting method does not fully solve the force field prediction problem since it relies on some DFT reference data. We attempted to employ the linear regression model to fit only the absolute DFT energy based on the same descriptor as used in reference [39]. The resulting MAE values are 944 meV atom$^{-1}$ and 6.329 eV Å$^{-1}$ for energy and force when the force coefficient is $10^{-4}$. The MAE values of formation energy fitting yield remarkable improvement of 22 meV atom$^{-1}$ and 0.243 eV Å$^{-1}$ for energy and force, respectively. Despite this improvement, the accuracy is insufficient. Perhaps, it is due to lack of fine tuning of hyperparameters, such as atomic weights and cutoff radii for each species.

To obtain a better accuracy, we decided to fit the absolute DFT energy and forces based on the NNP model. We employed the smooth SO3 power spectrum as the descriptor, which are formed by $n_{\max} = 4$ and $l_{\max} = 3$ with 40 components in total up to the cutoff radius of 5.0 Å$^{-1}$. The NNP training is executed with 2 hidden layers with 20 nodes for each layer while energy, force, and stress contributions are trained simultaneously. The importance coefficients of force and stress are set to $10^{-3}$ and $10^{-4}$, respectively. The results of the NNP training is illustrated in figure 3. The NNP energy MAE values for the training and test sets are 6.69 and 7.57 meV atom$^{-1}$, and the NNP force MAE values are 0.14 and 0.17 eV Å$^{-1}$. In addition, the MAE value of stress for training set is 0.078 GPa. Our results of energy and force yield worse performance compared to the previous report. Nevertheless, our NNP model offers a more general representation of the DFT PES since it does not rely on any prior reference values.

Furthermore, we calculated physical properties such as elastic constants, bulk and shear moduli, and the Poisson's ratio of the cubic elemental crystals (see table 2). From the table, the overall performances of NNP in predicting the physical properties are reasonable, except that the $C_{44}$ value of Nb is negative. However, this is consistent with the fact that the DFT's $C_{44}$ is also significantly lower than other terms. Hence, the negative

**Table 2.** Comparison of physical properties predicted with SO3-NNP. The DFT and experiment values are obtained from [39]. $B$ and $G$ denote the empirical Voigt–Reuss–Hill average bulk and shear moduli. $\nu$ is the Poisson's ratio. The DFT results were taken from open database of Materials Project [63].

| | $C_{11}$ | $C_{12}$ | $C_{44}$ | $B$ | $G$ | $\nu$ |
|---|---|---|---|---|---|---|
| | (GPa) | (GPa) | (GPa) | (GPa) | (GPa) | |
| Mo | | | | | | |
| DFT | 472 | 158 | 106 | 262 | 127 | 0.30 |
| [39] | 435 | 169 | 96 | 258 | 110 | 0.31 |
| NNP | **453** | **161** | **107** | **259** | **121** | **0.30** |
| Nb | | | | | | |
| DFT | 233 | 145 | 11 | 174 | 24 | 0.45 |
| [39] | 266 | **142** | **20** | 183 | **32** | 0.42 |
| NNP | **255** | 130 | −3 | **171** | N/A | **0.47** |
| Ta | | | | | | |
| DFT | 265 | 158 | 69 | 194 | 63 | 0.35 |
| [39] | **257** | **161** | **67** | **193** | 59 | 0.36 |
| NNP | 280 | 165 | 72 | 203 | **66** | **0.35** |
| W | | | | | | |
| DFT | 510 | 201 | 143 | 304 | 147 | 0.29 |
| [39] | 560 | 218 | 154 | 332 | 160 | **0.29** |
| NNP | **527** | **196** | **143** | **306** | **151** | **0.29** |

**Table 3.** The trained RMSE values of the predicted energy and forces of Pt dataset from the 30-40-40-1 NNP model. For reference, the results from the DeepPot-SE model [37] is also reported. It should be noted that that DeepPot-SE results were based on training the entire $MoS_2$/Pt dataset.

| | SO3-NNP | | DeepPot-SE [37] | |
|---|---|---|---|---|
| | Energy (meV) | Force (meV Å$^{-1}$) | Energy (meV) | Force (meV Å$^{-1}$) |
| Bulk | 1.64 | 64 | 2.00 | 84 |
| Surface | 5.91 | 87 | 6.77 | 105 |
| Cluster | 7.63 | 247 | 30.6 | 201 |

$C_{44}$ is acceptable if one considers the noise of stress data in training. Meanwhile, the $C_{44}$ value may be remedied by providing additional training dataset focusing on the shearing effect of Nb, increasing the importance of the stress coefficient, or increasing the hidden layer size in the NNP training. In addition, SO3 descriptor can be conveniently expanded in terms of both radial basis ($n_{\max}$) and angular momentum ($l_{\max}$) for achieving better overall accuracy.

### 5.3. Pt MLP for general purposes

Compared to crystalline systems, surfaces and nanoparticles generally represent the more challenging cases in MLP training as the nanoparticle contain more versatile atomic environments and more complex PES is expected. Here, we applied the NNP model to a Pt dataset [37], which consists of three data types: Pt surface, Pt bulk, and Pt cluster. There are 927 clusters of 15 atoms, and the Pt bulk type consists of 1717 configurations which are composed of 256 atoms. Pt surface are constructed from (001), (110), and (111) surfaces. Respectively, there are 949, 819, and 700 structures which consist of 320, 160, and 320 atoms. The SO3 power spectrum descriptor with $l_{\max} = 3$ and $n_{\max} = 4$ at radius cutoff of 4.9 Å$^{-1}$ was used to construct the MLP in the NNP model with two hidden layers with 30 nodes each. Unlike the previous examples, the minibatch scheme with the Adam optimizer was employed. In each iteration, the training process was updated in a batch size of 25 configurations.

In the original literature [37], DeepPot-SE includes $MoS_2$ slab and Pt clusters on $MoS_2$ substrate ($MoS_2$/Pt). The performance of DeepPot-SE yields satisfactory results. Meanwhile, embedded atom neural networks method can achieves outstanding results using a fraction of the same dataset [38]. Both of the methods exploit a large number of neural networks parameters, in the order of $10^4$–$10^5$, while the current study only adopts 2191 weight parameters for exemplary purpose. As shown in table 3, the accuracy from our small neural network model is comparable to that of DeepPot-SE results. Not surprisingly, the group of Pt bulk has the lowest errors with only 1.64 meV atom$^{-1}$ for the RMSE in energy and 67 meV Å$^{-1}$ in force.

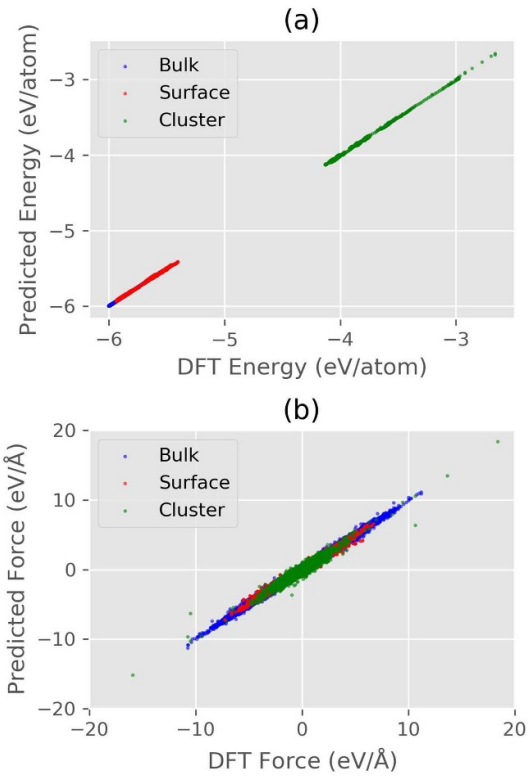


**Figure 4.** The correlation plots between NNP and DFT for (a) energy and (b) forces in the Pt system.

On the contrary, the errors on Pt clusters are about 2–4 time higher for both energy and forces. This is expected since the local atomic environments in the clusters are more diverse and thus learning the relation is harder. Nevertheless, the values from this exploratory study is comparable to the results from deep learning models. This example also suggests that a small NNP model with the properly constructed features can be a complementary solution for MLP development.

## 6. Conclusion

In conclusion, we introduced PyXtal_FF, a versatile package for developing MLPs that can perform at the DFT level. Currently, the code allows one to construct the MLPs from four different types of atom-centered descriptors: (w)ACSFs, EAD, SO4 bispectrum, or SO3 power spectrum. Two regression models, the generalized linear regression and neural networks, are supported to train the MLP by simultaneously fitting the data of energy, forces and stresses from the *ab-initio* simulation. In particular, we focus on the neural networks potential development. Our software package utilizes PyTorch as the main machinery, which is equipped with neural network models, automatic differentiation, as well as various optimization algorithms. We demonstrated the features of the current PyXtal_FF version by three examples on $SiO_2$, NbMoTaW HEA, and elemental Pt, respectively. In general, the mean absolute error values of each trained MLPs fall into the range of several meVs-atom$^{-1}$ in energy and several hundred meV Å$^{-1}$ in forces. While training on stress is optional, it is helpful to improve the general accuracy of the model. More importantly, this is crucial to yield better prediction on materials' elastic properties. As such, the MLPs can be applied to investigate the materials properties in greater accuracy than the classical potentials built from the empirical model. Finally, the PyXtal_FF is an open source code. We welcome anyone who is interested in MLP development to contribute to this project.

## Acknowledgment

We acknowledge the NSF (I-DIRSE-IL: 1940272 and 1940124) for their financial supports. The computing resources are provided by XSEDE (TG-DMR180040).

## Appendix A. The derivatives of ACSF

Following the equation (6) the derivative with respect to an atom $m$ can be written in the following form:

$$\frac{\partial G_i^{(2)}}{\partial \boldsymbol{r}_m} = \sum_{j\neq i} e^{-\eta(R_{ij}-R_s)^2}\left(\frac{\partial f_c}{\partial R_{ij}} - 2\eta(R_{ij}-R_s)f_c\right)\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}. \tag{A1}$$

For the periodic system, the computation of $\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}$ is straightforward except that one needs to consider one additional case. When $i=j$, the derivative is always zero:

$$\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m} = \begin{cases} 0 & m \notin [i,j], \\ 0 & m = i = j, \\ -\frac{\boldsymbol{r}_{ij}}{R_{ij}} & m = i \text{ (when } i \neq j), \\ \frac{\boldsymbol{r}_{ij}}{R_{ij}} & m = j \text{ (when } i \neq j) \end{cases} \tag{A2}$$

In equations (7) and (8), the cosine function can be defined as

$$\cos\theta_{ijk} = \frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ik}}{R_{ij}R_{ik}} \tag{A3}$$

where $\boldsymbol{r}_{ij}$ is the relative position between atom $j$ and atom $i$.

In the following, the expressions for the derivative with respect to an interacting atom $m$ are

$$\begin{aligned}\frac{\partial G_i^{(4)}}{\partial \boldsymbol{r}_m} =& 2^{1-\zeta}\sum_{j\neq i}\sum_{k\neq i,j} e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} \\ & \Bigg[\lambda\zeta(1+\lambda\cos\theta_{ijk})^{\zeta-1}\frac{\partial\cos\theta_{ijk}}{\partial \boldsymbol{r}_m}f_c(R_{ij})f_c(R_{ik})f_c(R_{jk}) - \\ & 2\eta(1+\lambda\cos\theta_{ijk})^{\zeta}\left(R_{ij}\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}+R_{ik}\frac{\partial R_{ik}}{\partial \boldsymbol{r}_m}+R_{jk}\frac{\partial R_{jk}}{\partial \boldsymbol{r}_m}\right) \\ & f_c(R_{ij})f_c(R_{ik})f_c(R_{jk}) \\ & +(1+\lambda\cos\theta_{ijk})^{\zeta}\left(\frac{\partial f_c(R_{ij})}{\partial R_{ij}}\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}f_c(R_{ik})f_c(R_{jk})\right. \\ & \left.+f_c(R_{ij})\frac{\partial f_c(R_{ik})}{\partial R_{ik}}\frac{\partial R_{ik}}{\partial \boldsymbol{r}_m}f_c(R_{jk})+f_c(R_{ij})f_c(R_{ik})\frac{\partial f_c(R_{jk})}{\partial R_{jk}}\frac{\partial R_{jk}}{\partial \boldsymbol{r}_m}\right)\Bigg]\end{aligned} \tag{A4}$$

$$\begin{aligned}\frac{\partial G_i^{(5)}}{\partial \boldsymbol{r}_m} =& 2^{1-\zeta}\sum_{j\neq i}\sum_{k\neq i,j} e^{-\eta(R_{ij}^2+R_{ik}^2)} \\ & \Bigg[\lambda\zeta(1+\lambda\cos\theta_{ijk})^{\zeta-1}\frac{\partial\cos\theta_{ijk}}{\partial \boldsymbol{r}_m}f_c(R_{ij})f_c(R_{ik}) \\ & -2\eta(1+\lambda\cos\theta_{ijk})^{\zeta}\left(R_{ij}\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}+R_{ik}\frac{\partial R_{ik}}{\partial \boldsymbol{r}_m}+R_{jk}\frac{\partial R_{jk}}{\partial \boldsymbol{r}_m}\right)f_c(R_{ij})f_c(R_{ik}) \\ & +(1+\lambda\cos\theta_{ijk})^{\zeta}\left(\frac{\partial f_c(R_{ij})}{\partial R_{ij}}\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}f_c(R_{ik})+f_c(R_{ij})\frac{\partial f_c(R_{ik})}{\partial R_{ik}}\frac{\partial R_{ik}}{\partial \boldsymbol{r}_m}\right)\Bigg].\end{aligned} \tag{A5}$$

The derivatives of atomic distances, $R_{ij}$ and $R_{ik}$, carry the same meaning as in equation (A2). The expression of the cosine of triple-atom angle is

$$\frac{\partial\cos\theta_{ijk}}{\partial \boldsymbol{r}_m} = \frac{\boldsymbol{r}_{ik}}{R_{ij}R_{ik}}\cdot\frac{\partial \boldsymbol{r}_{ij}}{\partial \boldsymbol{r}_m}+\frac{\boldsymbol{r}_{ij}}{R_{ij}R_{ik}}\cdot\frac{\partial \boldsymbol{r}_{ik}}{\partial \boldsymbol{r}_m}-\frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ik}}{R_{ij}^2R_{ik}}\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}-\frac{\boldsymbol{r}_{ij}\cdot\boldsymbol{r}_{ik}}{R_{ij}R_{ik}^2}\frac{\partial R_{ik}}{\partial \boldsymbol{r}_m} \tag{A6}$$

$$\frac{\partial \boldsymbol{r}_{ij}}{\partial \boldsymbol{r}_m} = \begin{bmatrix}\delta_{mj}-\delta_{mi} & 0 & 0 \\ 0 & \delta_{mj}-\delta_{mi} & 0 \\ 0 & 0 & \delta_{mj}-\delta_{mi}\end{bmatrix}, \tag{A7}$$

where $\delta_{mj}$ is the Kronecker delta between atom $m$ and $j$.

## Appendix B. The derivatives of EAD

The expression of the derivative with respect to an interacting atom $m$ is shown in the following:

$$\frac{\partial \rho_i}{\partial \boldsymbol{r}_m} = \sum_{l_x,l_y,l_z=0}^{l_x+l_y+l_z=L} \frac{2L_{\max}!}{l_x!l_y!l_z!}\left[\sum_{j\neq i}^{N} Z_j \Phi\right]\left[\sum_{j\neq i}^{N} Z_j \frac{\partial \Phi}{\partial \boldsymbol{r}_m}\right], \tag{B1}$$

where the derivative of $\Phi$ with respect to an interacting atom $m$ is

$$\begin{aligned}\frac{\partial \Phi}{\partial \boldsymbol{r}_m} =& \frac{e^{-\eta(R_{ij}-R_s)^2}}{R_c^{l_x+l_y+l_z}}\left[\left(\frac{\partial x_{ij}^{l_x}}{\partial \boldsymbol{r}_m} y_{ij}^{l_y} z_{ij}^{l_z} + x_{ij}^{l_x}\frac{\partial y_{ij}^{l_y}}{\partial \boldsymbol{r}_m} z_{ij}^{l_z} + x_{ij}^{l_x} y_{ij}^{l_y}\frac{\partial z_{ij}^{l_z}}{\partial \boldsymbol{r}_m}\right) f_c \right. \\ &\left. + x_{ij}^{l_x} y_{ij}^{l_y} z_{ij}^{l_z}\left(\frac{\partial f_c}{\partial R_{ij}} - 2f_c\eta(R_{ij}-R_s)\right)\frac{\partial R_{ij}}{\partial \boldsymbol{r}_m}\right].\end{aligned} \tag{B2}$$

## ORCID iDs

Howard Yanxon https://orcid.org/0000-0002-5078-1074
Qiang Zhu https://orcid.org/0000-0002-9892-0344

## References

[1] Yamakov V, Wolf D, Phillpot S R, Mukherjee A K and Gleiter H 2002 Dislocation processes in the deformation of nanocrystalline aluminium by molecular-dynamics simulation *Nat. Mater.* **1** 45–9
[2] Terrones M, Banhart F, Grobert N, Charlier J-C, Terrones H and Ajayan P 2002 Molecular junctions by joining single-walled carbon nanotubes *Phys. Rev. Lett.* **89** 075505
[3] Li X, Wei Y, Lu L, Lu K and Gao H 2010 Dislocation nucleation governed softening and maximum strength in nano-twinned metals *Nature* **464** 877–80
[4] Kresse G and Hafner J 1993 Ab initio molecular dynamics for liquid metals *Phys. Rev.* B **47** 558
[5] Kohn W and Sham L J 1965 Self-consistent equations including exchange and correlation effects *Phys. Rev.* **140** A1133
[6] Daw M S and Baskes M I 1984 Embedded-atom method: Derivation and application to impurities, surfaces and other defects in metals *Phys. Rev.* B **29** 6443
[7] Daw M S, Foiles S M and Baskes M I 1993 The embedded-atom method: a review of theory and applications *Mater. Sci. Rep.* **9** 251–310
[8] Tersoff J 1986 New empirical model for the structural properties of silicon *Phys. Rev. Lett.* **56** 632
[9] Stillinger F H and Weber T A 1985 Computer simulation of local order in condensed phases of silicon *Phys. Rev.* B **31** 5262
[10] MacKerell Jr A D *et al* 1998 All-atom empirical potential for molecular modeling and dynamics studies of proteins *J. Phys. Chem.* B **102** 3586–3616
[11] Behler J 2015 Constructing high-dimensional neural network potentials: A tutorial review *Int. J. Quantum Chem.* **115** 1032–50
[12] Artrith N and Behler J 2012 High-dimensional neural network potentials for metal surfaces: A prototype study for copper *Phys. Rev.* B **85** 045439
[13] Li W, Ando Y, Minamitani E and Watanabe S 2017 Study of Li atom diffusion in amorphous $Li_3PO_4$ with neural network potential *J. Chem. Phys.* **147** 214106
[14] Behler J and Parrinello M 2007 Generalized neural-network representation of high-dimensional potential-energy surfaces *Phys. Rev. Lett.* **98** 146401
[15] Behler J 2011 Atom-centered symmetry functions for constructing high-dimensional neural network potentials *J. Chem. Phys.* **134** 074106
[16] Artrith N, Morawietz T and Behler J 2011 High-dimensional neural-network potentials for multicomponent systems: applications to zinc oxide *Phys. Rev.* B **83** 153101
[17] Hajinazar S, Shao J and Kolmogorov A N 2017 Stratified construction of neural network based interatomic models for multicomponent materials *Phys. Rev.* B **95** 014114
[18] 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
[19] Bartók A P, Payne M C, Kondor R and Csányi G 2010 Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons *Phys. Rev. Lett.* **104** 136403
[20] Bartók A P, Kondor R and Csányi G 2013 On representing chemical environments *Phys. Rev.* B **87** 184115
[21] 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
[22] Shapeev A V 2016 Moment tensor potentials: a class of systematically improvable interatomic potentials *Multiscale Model. Simul.* **14** 1153–73
[23] Chen C, Deng Z, Tran R, Tang H, Chu I-H and Ong S P 2017 Accurate force field for molybdenum by machine learning large materials data *Phys. Rev. Mater.* **1** 043603
[24] Li X-G, Hu C, Chen C, Deng Z, Luo J and Ong S P 2018 Quantum-accurate spectral neighbor analysis potential models for ni-mo binary alloys and fcc metals *Phys. Rev.* B **98** 094104
[25] Szlachta W J, Bartók A P and Csányi G 2014 Accuracy and transferability of gaussian approximation potential models for tungsten *Phys. Rev.* B **90** 104108
[26] Deringer V L, Proserpio D M, Csányi G and Pickard C J 2018 Data-driven learning and prediction of inorganic crystal structures *Faraday Discuss.* **211** 45–59

[27] Deringer V L, Pickard C J and Csányi G 2018 Data-driven learning of total and local energies in elemental boron *Phys. Rev. Lett.* **120** 156001
[28] 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
[29] Singraber A, Morawietz T, Behler J and Dellago C 2019 Parallel multistream training of high-dimensional neural network potentials *J. Chem. Theory Comput.* **15** 3075–92
[30] Lee K, Yoo D, Jeong W and Han S 2019 Simple-nn: an efficient package for training and executing neural-network interatomic potentials *Comput. Phys. Commun.* **242** 95–103
[31] Khorshidi A and Peterson A A 2016 Amp: a modular approach to machine learning in atomistic simulations *Comput. Phys. Commun.* **207** 310–24
[32] Shao Y, Hellström M, Mitev P D, Knijff L and Zhang C 2020 Pinn: a Python library for building atomic neural networks of molecules and materials *J. Chem. Inf. Modeling* **60** 1184–93 pMID: 31935100
[33] Schütt K T, Sauceda H E, Kindermans P-J, Tkatchenko A and Müller K-R 2018 Schnet—a deep learning architecture for molecules and materials *J. Chem. Phys.* **148** 241722
[34] Artrith N and Urban A 2016 An implementation of artificial neural-network potentials for atomistic materials simulations: performance for TiO2 *Comput. Mater. Sci.* **114** 135–50
[35] Yanxon H, Zagaceta D, Wood B C and Zhu Q 2020 Neural network potential from bispectrum components: a case study on crystalline silicon *J. Chem. Phys.* **153** 054118
[36] Zagaceta D, Yanxon H and Zhu Q 2020 Spectral neural network potentials for binary alloys *J. Appl. Phys.* **128** 045113
[37] Zhang L, Han J, Wang H, Saidi W, Car R and Weinan E 2018 End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems *Advances in Neural Information Processing Systems* pp 4436–46
[38] Zhang Y, Hu C and Jiang B 2019 Embedded atom neural network potentials: Efficient and accurate machine learning with a physically inspired representation *J. Phys. Chem. Lett.* **10** 4962–7
[39] Li X-G, Chen C, Zheng H and Ong S P 2019 Unravelling complex strengthening mechanisms in the NbMoTaW multi-principal element alloy with machine learning potentials (arXiv:1912.01789)
[40] Himanen L, Jäger M O, Morooka E V, Canova] F F, Ranawat Y S, Gao D Z, Rinke P and Foster A S 2020 Dscribe: library of descriptors for machine learning in materials science *Comput. Phys. Commun.* **247** 106949
[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] Faber F, Lindmaa A, von Lilienfeld O A and Armiento R 2015 Crystal structure representations for machine learning models of formation energies *Int. J. Quantum Chem.* **115** 1094–101
[43] Huo H and Rupp M 2017 Unified representation of molecules and crystals for machine learning (arXiv:1704.06439)
[44] Singraber A, Behler J and Dellago C 2019 Library-based lammps implementation of high-dimensional neural network potentials *J. Chem. Theory Comput.* **15** 1827–40
[45] Gastegger M, Schwiedrzik L, Bittermann M, Berzsenyi F and Marquetand P 2018 WACSF–weighted atom-centered symmetry functions as descriptors in machine learning potentials *J. Chem. Phys.* **148** 241709
[46] Imbalzano G, Anelli A, Giofré D, Klees S, Behler J and Ceriotti M 2018 Automatic selection of atomic fingerprints and reference configurations for machine-learning potentials *J. Chem. Phys.* **148** 241730
[47] Daw M S and Baskes M I 1983 Semiempirical, quantum mechanical calculation of hydrogen embrittlement in metals *Phys. Rev. Lett.* **50** 1285
[48] Larsen A H *et al* 2017 The atomic simulation environment–a python library for working with atoms *J. Phys.: Condens. Matter.* **29** 273002
[49] Walt S V d, Colbert S C and Varoquaux G 2011 The numpy array: a structure for efficient numerical computation *Comput. Sci. Eng.* **13** 22–30
[50] Paszke A *et al* 2019 Pytorch: An imperative style, high-performance deep learning library in: Eds H Wallach, H Larochelle, A Beygelzimer, F d Alche-Buc, E Fox and R Garnett *Advances in Neural Information Processing Systems 32* (Red Hook, NY: Curran Associates, Inc.) pp 8024–35
[51] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L and Lerer A 2017 Automatic differentiation in PyTorch *NIPS 2017 Workshop*
[52] Liu D C and Nocedal J 1989 On the limited memory BFGS method for large scale optimization *Math. Program.* **45** 503–28
[53] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv: 1412.6980)
[54] Qian N 1999 On the momentum term in gradient descent learning algorithms *Neural Netw.* **12** 145–51
[55] Plimpton S 1995 Fast parallel algorithms for short-range molecular dynamics *J. Comp. Phys.* **117** 1–19
[56] Togo A and Tanaka I 2015 First principles phonon calculations in materials science *Scr. Mater.* **108** 1–5
[57] Hinuma Y, Pizzi G, Kumagai Y, Oba F and Tanaka I 2017 Band structure diagram paths based on crystallography *Comput. Mater. Sci.* **128** 140–84
[58] Matscipy (https://gitlab.com/libAtoms/matscipy)
[59] Kresse G and Furthmüller J 1996 Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set *Phys. Rev.* B **54** 11169–86
[60] Perdew J P, Burke K and Ernzerhof M 1996 Generalized gradient approximation made simple *Phys. Rev. Lett.* **77** 3865–8
[61] Yeh J-W, Chen S-K, Lin S-J, Gan J-Y, Chin T-S, Shun T-T, Tsau C-H and Chang S-Y 2004 Nanostructured high-entropy alloys with multiple principal elements: novel alloy design concepts and outcomes *Adv. Eng. Mater.* **6** 299–303
[62] Senkov O N, Wilks G, Scott J and Miracle D B 2011 Mechanical properties of $Nb_{25}Mo_{25}Ta_{25}W_{25}$ and $V_{20}Nb_{20}Mo_{20}Ta_{20}W_{20}$ refractory high entropy alloys *Intermetallics* **19** 698–706
[63] Jain A *et al* 2013 Commentary: the materials project: a materials genome approach to accelerating materials innovation *APL Mater.* **1** 011002