The Journal of Open Source Software

FitSNAP: Atomistic machine learning with LAMMPS

A. Rohskopf[1], **C. Sievers**[1,2], **N. Lubbers**[3], **M. A. Cusentino**[4], **J. Goff**[1], **J. Janssen**[3], **M. McCarthy**[1], **D. Montes de Oca Zapiain**[5], **S. Nikolov**[1], **K. Sargsyan**[6], **D. Sema**[7], **E. Sikorski**[1], **L. Williams**[6], **A. P. Thompson**[1], **and M. A. Wood**[1¶]

1 Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, United States of America **2** Boeing, Seattle, WA, United States of America **3** Los Alamos National Laboratory, Los Alamos, NM, United States of America **4** Material, Physical, and Chemical Sciences Center, Sandia National Laboratories, Albuquerque, NM, United States of America **5** Center for Integrated Nanotechnologies, Sandia National Laboratories, Albuquerque, NM, United States of America **6** Chemistry, Combustion and Materials Science Center, Sandia National Laboratories, Livermore, CA, United States of America **7** Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, United States of America ¶ Corresponding author

DOI: 10.21105/joss.05118

Software

- Review
- Repository
- Archive

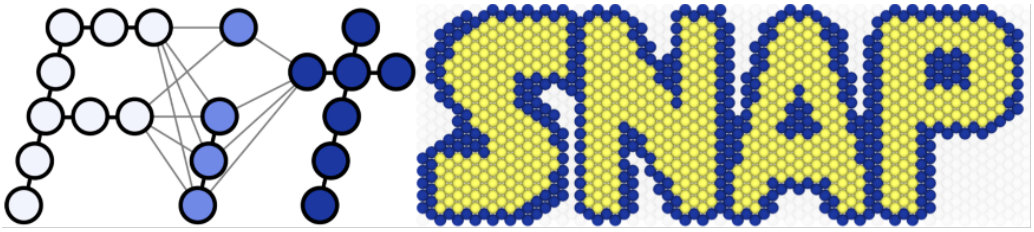
Editor: Jacob Schreiber

Reviewers:

- @tpurcell90
- @bahung

Submitted: 02 December 2022
Published: 27 March 2023

Summary



Chemical and physical properties of complex materials emerge from the collective motions of the constituent atoms. These motions are in turn determined by a variety of interatomic interactions mediated by the local redistribution of valence electrons about the fixed core electrons and nuclear charges. Scientific and engineering advances in materials science, chemistry, and many related fields benefit from our ability to directly sample the equilibrium and kinetic probability distributions of large collections of atoms and molecules. Classical molecular dynamics (MD) is a widely used simulation method in which Newton's equations of motion are numerically integrated forward in time to generate representative atomic trajectories of the atoms, from which insight into a wide range of material behaviors can be obtained. This simulation technique is not restricted to the definition of particles as atoms, but is generalizable to other types of interacting particles, such as coarse-grained beads in polymers, discrete elements representing granular materials, and fluid mass elements in dissipative particle dynamics. While all of these simulation methods use the same core algorithms as MD, we restrict our discussion here to the treatment of interactions between atoms.

Forces on atoms arise from the electronic structure, from electron charge densities that can be accurately calculated using quantum mechanical methods such as Density Functional Theory (DFT). While quantum mechanical methods (QM) are accurate, their computational cost scales at best as the third power of the number of electrons in the system, preventing application of the method to size and time scales relevant to many phenomena. It is therefore of great benefit to approximate the forces on atoms using simple empirical expressions for the energy of a configuration of atoms as a function of their local relative positions. From these functions, called interatomic potentials or just "potentials", the forces on atoms can be obtained by taking gradients with respect to atomic positions. Many of these empirical potentials are

derived from known physical and chemical interaction models, i.e. covalent, metallic, ionic, etc. By smoothly truncating these potentials at a suitable cutoff distance, the computational cost scales linearly in the number of atoms in the system and is amenable to efficient parallel algorithms available in MD codes such as LAMMPS (Thompson et al., 2022). As a result, these empirical models can be used to simulate systems that are far beyond the reach of QM. However, except for highly idealized structures, such as bulk crystals, these empirical models do not provide reliable surrogates for QM, and are prone to exhibiting unphysical behaviors. In the last decade, great progress has been made in constructing machine learning (ML) surrogates for QM potential energy functions. Large datasets of small atomic configurations with energies and forces evaluated using QM are used to train regression models that map local atomic environments to atomic energies and forces. By implementing these ML potentials in LAMMPS and running on large supercomputers, it is possible to simulate systems containing billions of particles with QM accuracy (Nguyen-Cong et al., 2021). However, the lack of general tools to facilitate this ML research limits widespread use and progress in the field. To address this need, we have developed a software package called FitSNAP. FitSNAP provides a general set of tools that can be used to train and test a wide range of atomistic simulation models.

Statement of need

FitSNAP is a Python package for constructing a wide variety of ML interatomic interaction models that map local atomic environments to energies, forces, stress or other quantities associated with atoms. This mapping is achieved by first calculating atomic environment feature vectors (descriptors), and then using these to learn a regression model. There are other high-quality atomistic ML software packages in this application space, such as DeepMD (Wang et al., 2018), n2p2 (Singraber et al., 2019), pacemaker (Bochkarev et al., 2022), NequIP (Batzner et al., 2022), Allegro (Musaelian et al., 2023), and GAP (Bartók et al., 2010). All of these provide excellent capabilities for their particular class of ML potentials, but lack the flexibility to accommodate a wide range of descriptors and regression models. Other software packages like amp (Khorshidi & Peterson, 2016), TorchANI (Gao et al., 2020), and ACEsuit (Ortner & Kermode, 2020) generalize the ML problem to models with multiple kinds of descriptors and model forms. After constructing a ML potential, a critical test of its viability is using it in large-scale production simulations. This ensures its stability and tests its ability to predict properties and phenomena that it was not explicitly trained on, colloquially referred to as extrapolation. To achieve this important step some of the aforementioned software packages provide excellent support for their respective models in large-scale molecular simulation packages like LAMMPS (Thompson et al., 2022).

A seamless interface with LAMMPS is where FitSNAP stands out; we use native components of LAMMPS to calculate inputs to our ML models, ensuring full consistency between the model produced in training and the model used in production MD simulations. In this regard FitSNAP acts as a multi-scale link between quantum and classical methods; a model is trained on fine-scale QM data and then seamlessly deployed in large-scale classical MD simulations on high-performance computing platforms. By using the same LAMMPS code to compute descriptors in both training and production simulations, we also reduce code duplication, and increase the rate of innovation by eliminating barriers to deploying new descriptors and models in large-scale simulations. While FitSNAP uses the Python interface to the LAMMPS library, it is not tightly integrated with LAMMPS. This allows greater flexibility and speed in development that would not be possible if it was fully integrated, given the much larger user base and diversity of use cases that LAMMPS must support. FitSNAP is therefore free to improve and grow independently of LAMMPS in the area of ML models, while at the same time maintaining full consistency with LAMMPS data structures and descriptor calculations. Our interface is achieved by using LAMMPS `compute` objects; these calculate quantities for a single configurations of atoms, without performing MD. The `compute` objects calculate the descriptors for our ML models, ensuring that descriptors used during training are identical to the descriptors used in performance-optimized LAMMPS production simulations. This also

allows performance improvements achieved in the LAMMPS production code to immediately speed up the training process, rather than having to replicate the code improvements in the training software. FitSNAP has already taken advantage of this intrinsic LAMMPS interface in a number of publications that performed large-scale simulations with innovative ML atomistic models (Cusentino et al., 2020, 2021; Nikolov et al., 2021, 2022; Wood & Thompson, 2018). LAMMPS supports a rapidly growing and diverse set of descriptors and ML model forms(Zuo et al., 2020). The interface ensures that all of these can be made accessible to FitSNAP users, with a small amount of extra glue code. This flexibility is paramount for achieving a general use ML potential software, since different descriptors and models are appropriate for different materials physics, different accuracy requirements, and different performance needs (Zuo et al., 2020). The modularity of FitSNAP components allows one to choose different models combined with different descriptors; this rapid prototyping can help users find the best atomistic ML model for their particular application.

Components

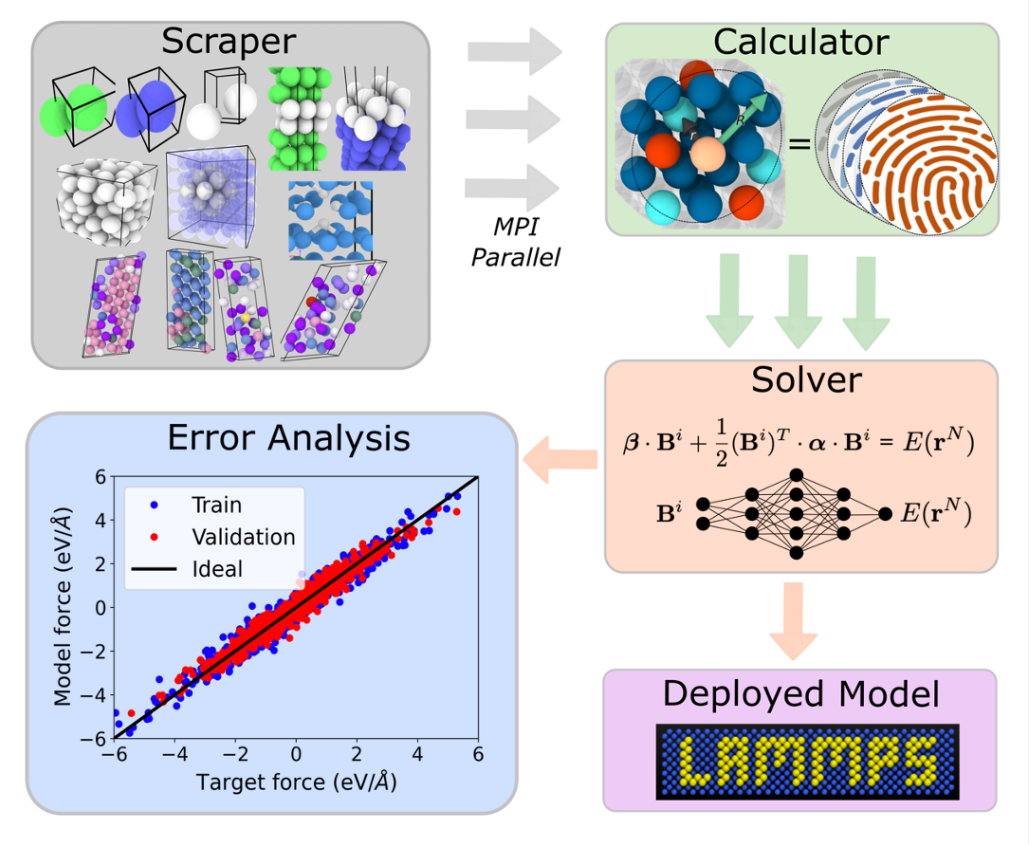


Figure 1: FitSNAP components and flow of control. The typical workflow involves scraping configurations of atoms which serve as training data; this is done in the `Scraper` class. Then we calculate ML features (atomic environment descriptors) in the `Calculator` class. Next, the ML problem is solved in the `Solver` class, which includes both linear and nonlinear models. This is followed by error analysis and/or model deployment in LAMMPS.

To generally approach the atomistic ML problem, we abstract components required for scraping data, calculating descriptors, and solving the optimization problem. The main software components driving this flow of control are shown in Figure 1. Scraping of training data and calculating atomic environment descriptors is parallelized over configurations of atoms with help

from our `ParallelTools` class. This stores data such as atomic positions, and fitting targets such as forces and energies, in shared memory arrays accessible to all processors on a compute node. This is achieved using the Python `mpi4py` package, allowing both (1) calculation of atomic environment descriptors in parallel across many configurations of atoms and (2) shared storage of training data including atomic environment descriptors and fitting targets. The latter point is vital for storing large amounts of data when using multiple processes on a single node; without shared arrays the required memory can easily exceed that available on many-core CPU platforms, since each processor would need to store all the data. The contents of these shared arrays are then accessible to the FitSNAP components on all processors in a node, throughout the rest of the workflow. The typical workflow begins with the Scraper class, which also uses `ParallelTools` to collect training data in parallel.

Scraper

The first step in the typical FitSNAP flow of control is a file I/O step to scrape the training data, the configurations of atoms and their associated energies, forces, spins, charges, or whatever fitting quantity; this is achieved with the `Scraper` class. Training data includes basic structural information about the set of atoms such as Cartesian coordinates and DFT simulation box vectors. Ground truth values used in the loss function during the regression step are collected at this stage; this "scraping" occurs in parallel using MPI. Accepted file formats currently include XYZ and JSON files, which are commonly used in the atomistic modelling community. These files are stored in directories in a manner determined by the user, where each directory can designate a FitSNAP data `group`; each group of configurations can receive its own training/testing fractions and fitting target weights, offering flexibility in how different configurations of atoms are weighted or tested during training. When scraping, the data is stored in a FitSNAP data dictionary that houses positions of atoms and their associated fitting quantities. While the typical flow of control involves first scraping training data from these files, using FitSNAP in library mode allows one to bypass this step if the training data is collated in some other manner, such as with the Atomistic Simulation Environment (ASE), stored in RAM, and inserted into the FitSNAP data dictionary that houses training data. Nonetheless, the FitSNAP data dictionary contains atomic positions that are then converted to atomic environment descriptors in the `Calculator` class.

Calculator

To transform this structural information into physically appropriate models, we employ the `Calculator` class which provides permutation, translationally and rotationally invariant descriptors. These descriptors can be calculated from their performant implementations in LAMMPS, and then extracted via the LAMMPS Python/C API, which inserts descriptors into the shared memory arrays of `ParallelTools`. For descriptors that are not implemented in LAMMPS, we provide a `Custom Calculator` class that extracts periodic-boundary-transformed LAMMPS positions which can be used to calculate custom coded descriptors in Python. This ensures extensibility of different descriptors for describing atomic environments. Regardless of the method of descriptor calculation, these calculations are parallelized via MPI over configurations of atoms stored in the FitSNAP data dictionary. Currently we include Spectral Neighbor Analysis Potential (SNAP) (Thompson et al., 2015) and Atomic Cluster Environment (ACE) (Drautz, 2019) descriptors which are both calculated in LAMMPS.

Solver

After collating the necessary descriptors and their target fitting quantities, the `Solver` class can either proceed to regression of the ML problem or evaluating the model. The connection between descriptor calculation, which can happen in LAMMPS, and FitSNAP solvers is exemplified in Figure 2 for a neural network potential inputting arbitrary atom-centered descriptors. In the typical flow of control for fitting a potential, `Solver` creates a loss function measuring difference between target and model fitting quantities such as energies and forces. This loss function

is then minimized with a method depending on the choice of user input. For linear models, solver types include singular value decomposition (SVD) or adaptive rectangular decomposition (ARD). One advantage of linear models is the possibility to analytically determine uncertainties in fitting coefficients with Bayesian statistics. FitSNAP therefore also includes uncertainty quantification (UQ) solvers, which output model coefficient covariances for linear models.

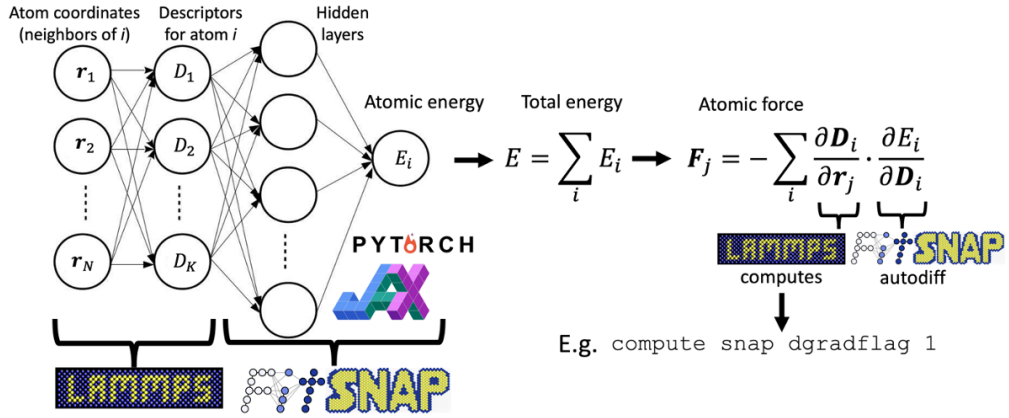


Figure 2: LAMMPS-FitSNAP interface for calculating energies and forces with machine learned potentials, illustrated here with a neural network potential that calculates atomic energies. Descriptor and descriptor gradient calculations occur in LAMMPS, while model and model gradient calculations occur with automatic differentiation (autodiff) frameworks in FitSNAP.

For nonlinear models, solver types include neural networks implemented in PyTorch or JAX. The set of available solver type depends on the form of the ML regression model. To ensure good computational performance when fitting nonlinear models to forces, we use a modified form of iterated back-propagation (Smith et al., 2020) where forces are calculated as a first back-propagation with respect to model inputs, but descriptor gradients are computed with their performant implementations in LAMMPS. This requires matching or aligning between descriptor gradients extracted from LAMMPS, and model gradients extracted from ML frameworks like PyTorch or JAX. To achieve this, we routinely verify the correctness of our forces by comparing them with finite difference estimates as a part of our continuous integration testing. Ultimately, after optimizing the ML model with the `Solver` class, FitSNAP produces LAMMPS-ready files which may be directly used as inputs to large-scale LAMMPS simulations.

Extensibility

The modularity of these components allows for different `Calculator` and `Solver` sub-classes, allowing for a variety of different descriptor and model combinations. Any descriptor may be programmed in the `Calculator` class, which can then be input to any ML model in the `Solver` class to perform fitting. This flexibility can also extend to model architectures entirely different to the commonly used atom-centered networks, such as pairwise networks (Jose et al., 2012) which are also included in FitSNAP. These custom and less traditional architectures are readily implemented using LAMMPS periodic-boundary-condition transformed positions. Aside from flexibility in choosing or implementing various descriptors and models, we also provide a library mode where a FitSNAP object can be created in any external Python script. This allows users to access internal methods and break the usual sequential flow of control for their specific applications. Some users for example may want to extract descriptors for a set of configurations and perform statistical analysis on that data, access hyperparameters in an external optimization tool, or even code an entirely new component not mentioned here. This Python library also allows interface to custom workflow frameworks like `pyiron` (Janssen et

al., 2019), allowing users to save, share, and modify their workflows for training and using machine learned potentials. In all cases, the ability to program and use any descriptor or model based on position data extracted from LAMMPS ensures utmost extensibility to future atomic environment descriptors and models, while allowing users to enjoy connection to a high-performance MD engine immediately after training potentials.

Funding Statement

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

All authors acknowledge funding support from the U.S. Department of Energy, Office of Fusion Energy Sciences (OFES) under Field Work Proposal Number 20-023149 and the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

Bartók, A. P., Payne, M. C., Kondor, R., & Csányi, G. (2010). Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, *104*(13), 136403. https://doi.org/10.1103/physrevlett.104.136403

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., & Kozinsky, B. (2022). E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, *13*(1), 1–11. https://doi.org/10.21203/rs.3.rs-244137/v1

Bochkarev, A., Lysogorskiy, Y., Menon, S., Qamar, M., Mrovec, M., & Drautz, R. (2022). Efficient parametrization of the atomic cluster expansion. *Physical Review Materials*, *6*(1), 013804. https://doi.org/10.1103/physrevmaterials.6.013804

Cusentino, M. A., Wood, M. A., & Thompson, A. P. (2020). Explicit multielement extension of the spectral neighbor analysis potential for chemically complex systems. *The Journal of Physical Chemistry A*, *124*(26), 5456–5464. https://doi.org/10.1021/acs.jpca.0c02450.s001

Cusentino, M. A., Wood, M. A., & Thompson, A. P. (2021). Beryllium-driven structural evolution at the divertor surface. *Nuclear Fusion*, *61*(4), 046049. https://doi.org/10.1088/1741-4326/abe7bd

Drautz, R. (2019). Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, *99*(1), 014104. https://doi.org/10.1103/physrevb.99.014104

Gao, X., Ramezanghorbani, F., Isayev, O., Smith, J. S., & Roitberg, A. E. (2020). TorchANI: A free and open source PyTorch-based deep learning implementation of the ANI neural

network potentials. *Journal of Chemical Information and Modeling*, *60*(7), 3408–3415. https://doi.org/10.26434/chemrxiv.12218294

Janssen, J., Surendralal, S., Lysogorskiy, Y., Todorova, M., Hickel, T., Drautz, R., & Neugebauer, J. (2019). Pyiron: An integrated development environment for computational materials science. *Computational Materials Science*, *163*, 24–36. https://doi.org/10.1016/j.commatsci.2018.07.043

Jose, K. J., Artrith, N., & Behler, J. (2012). Construction of high-dimensional neural network potentials using environment-dependent atom pairs. *The Journal of Chemical Physics*, *136*(19), 194111. https://doi.org/10.1063/1.4712397

Khorshidi, A., & Peterson, A. A. (2016). Amp: A modular approach to machine learning in atomistic simulations. *Computer Physics Communications*, *207*, 310–324. https://doi.org/10.1016/j.cpc.2016.05.010

Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C. J., Kornbluth, M., & Kozinsky, B. (2023). Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, *14*(1), 579. https://doi.org/10.1038/s41467-023-36329-y

Nguyen-Cong, K., Willman, J. T., Moore, S. G., Belonoshko, A. B., Gayatri, R., Weinberg, E., Wood, M. A., Thompson, A. P., & Oleynik, I. I. (2021). Billion atom molecular dynamics simulations of carbon at extreme conditions and experimental time and length scales. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–12. https://doi.org/10.1145/3458817.3487400

Nikolov, S., Tranchida, J., Ramakrishna, K., Lokamani, M., Cangi, A., & Wood, M. (2022). Dissociating the phononic, magnetic and electronic contributions to thermal conductivity: A computational study in alpha-iron. *Journal of Materials Science*, 1–14. https://doi.org/10.1007/s10853-021-06865-3

Nikolov, S., Wood, M. A., Cangi, A., Maillet, J.-B., Marinica, M.-C., Thompson, A. P., Desjarlais, M. P., & Tranchida, J. (2021). Data-driven magneto-elastic predictions with scalable classical spin-lattice dynamics. *Npj Computational Materials*, *7*(1), 1–12. https://doi.org/10.1038/s41524-021-00617-2

Ortner, C., & Kermode, J. (2020). ACEsuit. *GitHub Repository*. https://github.com/ACEsuit

Singraber, A., Behler, J., & Dellago, C. (2019). Library-based LAMMPS implementation of high-dimensional neural network potentials. *Journal of Chemical Theory and Computation*, *15*(3), 1827–1840. https://doi.org/10.1021/acs.jctc.8b00770.s001

Smith, J. S., Lubbers, N., Thompson, A. P., & Barros, K. (2020). Simple and efficient algorithms for training machine learning potentials to force data. *arXiv Preprint arXiv:2006.05475*. https://doi.org/10.2172/1763572

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., Veld, P. J. in't, Kohlmeyer, A., Moore, S. G., Nguyen, T. D., & others. (2022). LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, *271*, 108171. https://doi.org/10.1016/j.cpc.2021.108171

Thompson, A. P., Swiler, L. P., Trott, C. R., Foiles, S. M., & Tucker, G. J. (2015). Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, *285*, 316–330. https://doi.org/10.1016/j.jcp.2014.12.018

Wang, H., Zhang, L., Han, J., & Weinan, E. (2018). DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, *228*, 178–184. https://doi.org/10.1016/j.cpc.2018.03.016

Wood, M. A., & Thompson, A. P. (2018). Extending the accuracy of the SNAP interatomic potential form. *The Journal of Chemical Physics*, *148*(24), 241721. https://doi.org/10.1063/1.5017641

Zuo, Y., Chen, C., Li, X., Deng, Z., Chen, Y., Behler, J., Csányi, G., Shapeev, A. V., Thompson, A. P., Wood, M. A., & others. (2020). Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, *124*(4), 731–745. https://doi.org/10.1021/acs.jpca.9b08723.s001