Supporting Information

Addressing uncertainty in atomistic machine learning

Andrew Peterson[1,*], Rune Christensen[2], Alireza Khorshidi[1]

[1]*School of Engineering, Brown University, Providence, Rhode Island, 02912, United States.*
[2]*Technical University of Denmark.*
**andrew_peterson@brown.edu*

This supporting information contains background information on the bootstrap resampling procedure employed, a strategy to grow a bootstrap ensemble as new training images become available, and calculation details for the two examples discussed in the manuscript.

S-1 The bootstrap ensemble

The bootstrap resampling technique [1] involves randomly resampling a set of data in order to infer statistics about the population. Consider a sample set of data, such as

$$\mathbf{X} = [A, B, C, D, E] \tag{1}$$

In our case, A–E are individual atomic images, each containing information about atomic coordinates, potential energy, and forces, and $\mathbf{X}$ is the set of all training images available for atomistic machine learning. Bootstrapping simply involves resampling this set into an ensemble of sets; each new set is assembled by drawing individual data points from $\mathbf{X}$ with replacement. (Each resampled set has the same length as $\mathbf{X}$.) For example, we might draw new samples $\mathbf{X}_i^*$ such as:

$$\begin{aligned} \mathbf{X}_1^* &= [D, E, B, E, A] \\ \mathbf{X}_2^* &= [D, D, C, C, E] \\ \mathbf{X}_3^* &= [B, B, C, D, A] \\ &\dots \end{aligned} \tag{2}$$

In the limit of a large number of images in the sample, each image has a chance of about $1/e \approx 0.37$ of being left out of any individual new set. (That is, an image like C should appear in approximately 63% of the new sets $\mathbf{X}_i^*$.)

As described by Heskes [2], this can be applied to neural network regression in order to infer prediction intervals of the model on the population. However, as we describe in the text, we do not expect to be able to construct rigorous prediction intervals due to the strong likelihood that atomistic machine-learning models will be used outside of the region of the potential energy surface (PES) in which the training data was generated; that is, we should assume that users' data sets will not in general adequately sample the population. Because of this, we do not report rigorous prediction intervals in the text, but rather use simple distribution information from the ensemble prediction, like the halfspread of the data (defined in the text).

In our implementation, we construct an ensemble of training sets ($\mathbf{X}_1^*$, $\mathbf{X}_2^*$, $\mathbf{X}_3^*$, $\cdots$ $\mathbf{X}_N^*$), and use each training set ($\mathbf{X}_i^*$) to train an independent machine-learning calculator. (Structures of the machine-learning calculators for each example are described below.) Each calculator is initialized with random guesses for the neural network weights and trained to the same convergence value. Note that in the event that a calculator fails to train successfully (*i.e.*, it gets "stuck" in a local minimum on the loss-function surface), it is important that the replacement calculator is trained to the identical training set. If this is not done, it can result in the exclusion of certain training points (or combinations of points) that are difficult to train, perhaps representing regions of high curvature on the PES. The resulting collection of trained calculators is the ensemble of calculators reported on in the text. We can examine the spread between calculator predictions of energies or forces (that is, the disagreement between calculators), to give an indication of the confidence of the results, and as discussed in the text, a comparison of the ensemble (half)spread to the residuals of the test data can be used to evaluate the performance of the ensemble approach.

Accelerated re-training. Training of atomistic machine-learning calculators can be a computationally expensive undertaking, so training an ensemble of calculators adds considerable expense to this process. As noted in the text, we anticipate that the halfspread calculations should best be interpreted as an indication of if a new training point should be generated (that is, if the halfspread on a prediction is significantly higher than those in the training set, one should perform a new electronic structure calculation and add it to the training data). However, if this triggers a re-training event "from scratch", the re-training may be more computationally expensive than the electronic structure calculation itself.[1] For this reason, we suggest an algorithm by which the existing ensemble can be modified to incorporate a new image (or new images):

1. Add the new image to the parent training set. *E.g.*, equation (1) becomes

$$\mathbf{X} = [A, B, C, D, E, \underline{F}]. \tag{3}$$

   (In this and the following lists, the new items are underlined for clarity.) Let n be the new length of $\mathbf{X}$.

2. For each resampled set, add an image that is randomly chosen from the new set $\mathbf{X}$, such that each $\mathbf{X}_i^*$ becomes of length n. For example, equation (2) might become

$$\begin{aligned} \mathbf{X}_1^* &= [D, E, B, E, A, \underline{D}] \\ \mathbf{X}_2^* &= [D, D, C, C, E, \underline{F}] \\ \mathbf{X}_3^* &= [B, B, C, D, A, \underline{C}] \\ &\cdots \end{aligned} \tag{4}$$

[1]Of course, this depends on the type of electronic structure calculation as well as the size of the machine-learning models and training sets.

3. For each of the first $(n-1)$ elements of each resampled set $\mathbf{X}_i^*$, replace them with the new image (here: F) with probability $1/n$. This is equivalent to the probability of the new image being in any particular location if it were drawn from scratch. *E.g.*, equation (4) might become

$$\begin{aligned} \mathbf{X}_1^* &= [D, \underline{F}, B, E, A, D] \\ \mathbf{X}_2^* &= [D, D, C, C, E, F] \\ \mathbf{X}_3^* &= [B, \underline{F}, C, D, \underline{F}, C] \\ &\dots \end{aligned} \tag{5}$$

4. Re-train each calculator in the ensemble to its new respective data set, starting from the final converged parameters. Importantly, train not just to a root-mean square-error (RMSE) for the entire collection of images, but to a maximum tolerated residual on any individual image in the data set. This ensures that the model will fit the new image; if this is not done and the training set is large, the RMSE may be barely affected by the presence of the new image. Note that the loss function will still include higher weighting for redundant images in the set, so this aspect of bootstrapping should be preserved. In most cases, we expect that the re-training should be faster than the initial training, as the calculator is already in a position of the (original) loss function surface that results in a low error for most of the images. After re-training, the new ensemble is complete.

S-2 Calculation details for water molecule

Parent data. The atomic structures of the water molecule were prepared in the Atomic Simulation Environment (ASE) [3], which provides interfaces to the NWChem calculator [4] and the *Amp* atomistic machine-learning package [5]. The water structures varied in an O–H bondlength, r/r_0, and the H–O–H bond angle, θ, where r_0 is the equilibrium bond length of a water molecule, taken as 0.969 Å. The other O–H bond was fixed at r_0. r/r_0 was varied at 20 values between 0.9 and 2.0, while θ/π was varied at 20 values between 0.5 and 1.0, with θ expressed in radians. The potential energy and forces were found for each static configuration by hybrid density functional theory in the NWChem calculator with the B3LYP exchange–correlation functional [6, 7] and the 6-31+G* basis set [8]. The trajectory of all configurations of the water molecule (along with energy and forces) is included as a supporting information file in the extended xyz (.extxyz) format, which as a standardized plain-text file is both machine-parsable and human readable.

Machine learning. The open-source *Amp* project [5,9], of which we are developers, was used for the atomistic machine learning. This software is freely available[2] and is designed to integrate with ASE, making it simple to interface with standard atomistic methods such as geometry optimization algorithms or molecular dynamics methods, and giving it access to a wide range of electronic structure calculators. To fit each member of the committee, we used the Gaussian descriptor class

[2]The package can be downloaded from its repository at https://bitbucket.org/andrewpeterson/amp; its documentation is hosted at http://amp.readthedocs.io. An archival version is available via reference [9].

with a 6.5 Å cutoff radius with the code-default set of symmetry functions, similar to the form described by Behler and Parrinello [10]. In this scheme, two classes of Gaussian-type descriptors are used. The first accounts for radial interactions about atom i with neighboring atoms $j \neq i$ (for neighbors within a cutoff radius R_c of atom i) as

$$G_i^\mathrm{I} = \sum_{j \neq i} e^{-\eta R_{ij}^2 / R_\mathrm{c}^2} f_\mathrm{c}(R_{ij}) \tag{6}$$

R_{ij} is the distance between atoms i and j. η is a parameter that varies the strength of the interaction, and to make several components of the feature vector, several values of η ($\eta = 0.05, 4, 20, 80$) are employed. Each G^I is specific to interactions with a specific element type, so in this case four G^I elements for each of the two chemical species (a total of eight elements) enter the feature vector. Angular symmetry functions are also employed, which account for the angle $\theta_{ijk} = \cos^{-1}(\mathbf{R}_{ij} \cdot \mathbf{R}_{ik}/(R_{ij}R_{ik}))$ between atoms i, j, and k (where $\mathbf{R}_{ij}$ is the vector connecting atom i to atom j and $\mathbf{R}_{ik}$ connecting atom i to atom k, and R_{ij} and R_{ik} are the corresponding distances). These symmetry functions are computed as

$$G_i^\mathrm{II} = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos \theta_{ijk})^\zeta \, e^{-\eta \left(R_{ij}^2 + R_{ik}^2 + R_{jk}^2\right)/R_\mathrm{c}^2} f_\mathrm{c}(R_{ij}) f_\mathrm{c}(R_{ik}) f_\mathrm{c}(R_{jk}) \tag{7}$$

Here, the parameters ζ (=1, 4), η (=0.005) and λ (=+1, -1) are employed to make several values of the symmetry functions. Like in G^I, each G^II is specific to element (pair) types, so with possible pairs of (H, H), (O, H), and (O, O) (for j, k) an additional 12 elements are added to the feature vector, although in practice all (O, O) symmetry functions will be zero in this case of only an isolated water molecule. In total, this made each atom's feacture vector contain 20 elements.

A neural network was used (for each element type O and H) for the machine-learning regression with a structure of (20, 5, 5, 1); that is, with two layers of five hidden nodes. The combination of a Gaussian descriptor and a neural network backend, in atom-centered mode, is known as the Behler–Parrinello scheme [10]. In energy-only training mode, each model was regressed until the RMSE per atom was below 8.57 $\times$ 10^{-5} eV per atom; this was designed to be approximately the same as values encountered when both energy and forces were trained simultaneously. (Including forces in the regression typically leads to a tighter fit to energy as well; thus we specified this particular energy tolerance such that the two approaches could be directly compared in the text.) When energy and forces were used for training simultaneously, the forces were trained to an RMSE of 0.001 eV/Å, which is already a per-atom quantity. In constructing the loss function which controls the path of convergence, a coefficient of 0.04 was used to add the force residuals to the energy residuals.

Bootstrap ensemble. A selection of 27 images, as described in the text, were chosen to serve as the training images. An ensemble of 50 independent calculators was assembled from these images, as described in Section S-1. Each calculator was independently trained, using randomly initialized parameters, in the manner described in the previous paragraph. This ensemble of calculators was then used to provide median predictions and prediction intervals.

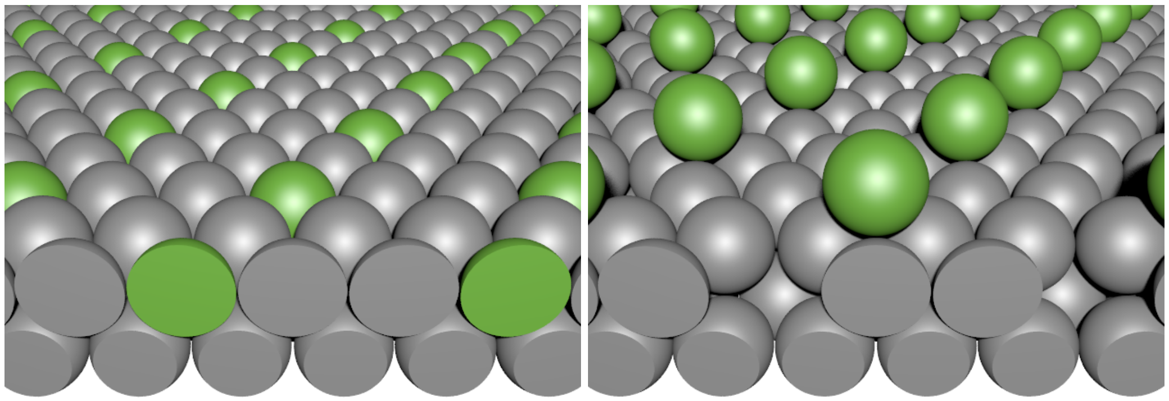

Figure S1: Pt fcc (111) system for the second example. On the left is the system before the defect is created; the right shows the created defect. The atom that moves is highlighted in green for clarity.

S-3 Calculation details for defect creation

Parent data: trajectory creation and electronic structure calculations. The creation of a vacancy is a rare event, so it is simpler to create a molecular dynamics simulation with the more common event of a vacancy being destroyed, then reverse its direction. To do this, we used ASE [3] to create a small Pt fcc(111) slab with two layers of six atoms per layer (in a $2\times3\times2$ arrangement, with the bottom layer fixed). A single atom (#11, with numbering starting at 0) was moved from its lattice position to a 3-fold fcc site on the surface. (An fcc site is a three-fold site with no atom directly below it in the second layer; conversely an hcp site contains an atom in this location.) The atomic system is shown in Figure S1. We used the experimental lattice constant and ran Langevin dynamics as implemented in ASE for 10,000 steps (at timesteps of 5 fs) in each simulation at increasing temperatures until we saw the adatom fill the vacancy, which occurred in these stochastic simulations at 2000 K. To facilitate rapid trajectory creation, the initial trajectories were created using the effective medium theory (EMT) [11, 12] calculator as implemented in ASE; the resulting images were then repeated in planewave density functional theory (DFT). Since in this example we are not concerned with the specific process of defect creation, but only in having representative images from before, after, and during this process, this saves considerable computational effort, particularly since this allowed us to parallelize the subsequent DFT calculations. The DFT calculations were conducted in Dacapo [3], a planewave calculator, with Vanderbilt pseudopotentials [13] and a dipole correction employed in the vacuum between adjacent slabs. The RPBE functional of Hammer, Hansen, and Nørskov [14] was used to account for exchange and correlation of electrons. The 10,000-image dynamical trajectory is included as a separate supporting information file.

Machine learning. The machine learning was again undertaken in *Amp* [5] using an identical approach as to that described in Section S-2, except with a default energy RMSE convergence value of 0.001 eV/atom, since no force training was applied in this example. Of course, the feature vectors in this example were smaller (four G^{I} elements and four G^{II} elements) because only a single chemical element is present in the simulations.

Bootstrap ensemble. A selection of 100 images was randomly sampled from the first 6,000 images of the 10,000-image trajectory. This is from the region from before the vacancy was created, which occurred at about image 7,071 as determined by the change in z coordinate of atom #11. These 100 images were used to train a 50-member bootstrap ensemble, in a method identical to that described in Section S-2. These 100 training images are included as an additional supporting information file with this publication.

References

[1] Efron, B. Bootstrap Methods: Another Look at the Jacknife. *The Annals of Statistics* **1979**; 7, 1–26.

[2] Heskes, T. Practical confidence and prediction intervals. In M.C. Mozer; M.I. Jordan; T. Petsche, eds., *Advances in Neural Information Processing Systems 9*. MIT Press. 1996.

[3] Bahn, S.R.; Jacobsen, K.W. An object-oriented scripting interface to a legacy electronic structure code. *Computing in Science & Engineering* **2002**; 4, 56–66.

[4] Valiev, M.; Bylaska, E.; Govind, N.; Kowalski, K.; Straatsma, T.; Dam, H.V.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T.; de Jong, W. NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications* **2010**; 181, 1477 – 1489.

[5] Khorshidi, A.; Peterson, A.A. *Amp*: A modular approach to machine learning in atomistic simulations. *Computer Physics Communications* **2016**; 207, 310 – 324.

[6] Becke, A.D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A* **1988**; 38, 3098–3100.

[7] Lee, C.; Yang, W.; Parr, R.G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B* **1988**; 37, 785–789.

[8] Ditchfield, R.; Hehre, W.J.; Pople, J.A. Self-Consistent Molecular-Orbital Methods. IX. An Extended Gaussian-Type Basis for Molecular-Orbital Studies of Organic Molecules. *The Journal of Chemical Physics* **1971**; 54, 724–728.

[9] Khorshidi, A.; Ulissi, Z.; El Khatib, M.; Peterson, A. Amp: The Atomistic Machine-learning Package v0.5. *Zenodo* **2017**; DOI:10.5281/zenodo.322427.

[10] Behler, J.; Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **2007**; 98, 146401.

[11] Jacobsen, K.; Stoltze, P.; Nørskov, J. A semi-empirical effective medium theory for metals and alloys. *Surface Science* **1996**; 366, 394 – 402.

[12] Nørskov, J.K.; Lang, N.D. Effective-medium theory of chemical binding: Application to chemisorption. *Phys. Rev. B* **1980**; 21, 2131–2136.

[13] Vanderbilt, D. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. *Physical Review B* **1990**; 41, 7892–7895.

[14] Hammer, B.; Hansen, L.B.; Nørskov, J.K. Improved adsorption energetics within density-functional theory using revised Perdew-Burke-Ernzerhof functionals. *Physical Review B* **1999**; 59, 7413–7421.