Tutorial of open source neural network applications

Jacob Boes

January 22, 2016

1 Introduction

All enclosed documentation and data is available from Github at: https://github.com/jboes/amp-tutorial.git

- Similarly, the PDF can be found here: https://github.com/jboes/amp-tutorial/blob/master/workbook.pdf
- All of the code described within is open source with the exception of the Vienna *ab initio* simulation package (VASP).
- Have questions or are interested in learning more? Feel free to contact me at: jboes@andrew.cmu.edu.

2 Theory

2.1 Density Functional Theory

In the Kitchin group, we use the VASP suit to perform Density functional theory (DFT) calculations.

- Solves the Kohn-Sham Equations which approximate the Schrodinger equation with electron density.

- VASP is ideal for bulk system since it uses plane waves to estimate the electron density.

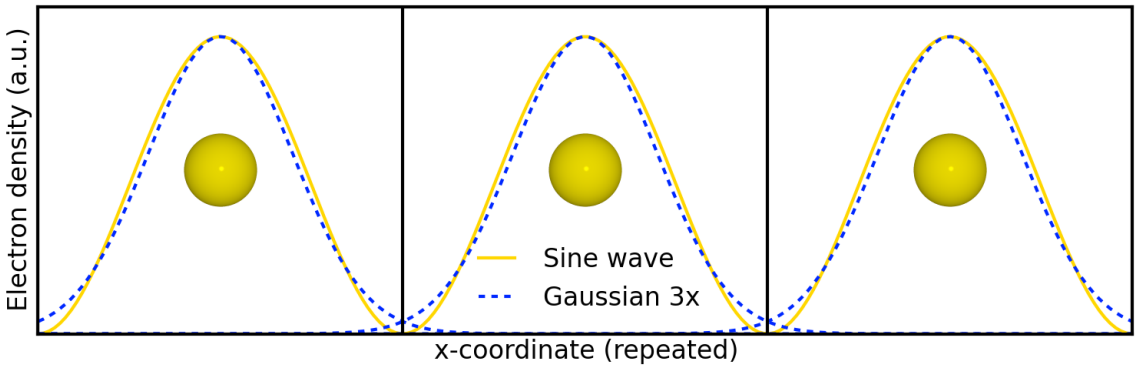


Figure 1: Example of a 2D electron density distribution using plane waves and Gaussians.

- Although powerful and accurate DFT is too slow for more advanced application, such as molecular dynamics (MD) and larger unit cells.
- MD requires a large number of calculations in series, which DFT is poorly suited for.

2.2 Neural Networks (NN)

Neural networks are a machine learning technique which can be use to construct a flexible function from an arbitrary input variable.

- This has been successfully implemented to predict potential energies (function output) from atomic positions (input variable).
- A basic feed-forward neural networks is demonstrated in Figure 2 for a 2 atom system.
- Feed-forward neural networks are limited by their construction. Atoms cannot be added or disordered using this method.

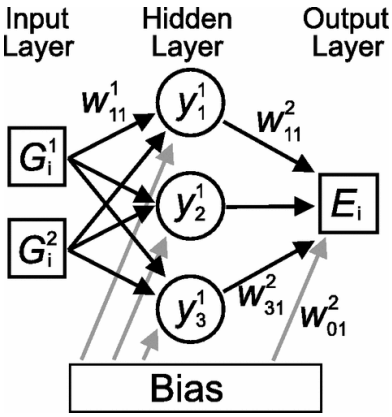


Figure 2: A basic neural network framework for a 2 atom system.

2.3 Behler-Parrinello descriptors

To use NNs on variable systems of atoms we need a different set of inputs than atomic positions.

- There are many ways to describe various atomic configurations other than Cartesian coordinates.
- Whatever descriptor we use needs to be accessible without performing a DFT calculation.
- Behler and Parrinello suggested a cutoff radius and "symmetry functions".

The cutoff radius R_c is useful because it limits the size of the symmetry function required. This is demonstrated in Figure 3.

- Choice of R_c is critical! We assume no interactions occur beyond this cutoff.
- This is not suitable for systems with long-range interactions.

For each atom, we construct a "symmetry function" made of various Behler-Parrinello descriptors. Some of these descriptors are demonstrated in Figure 4.

- We can use as many descriptors as needed to define the system.

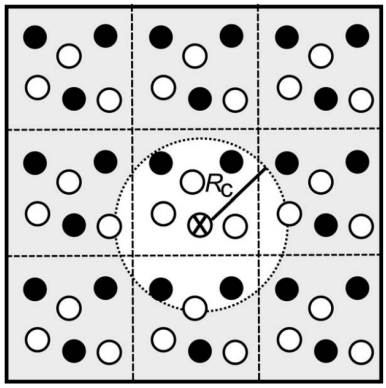
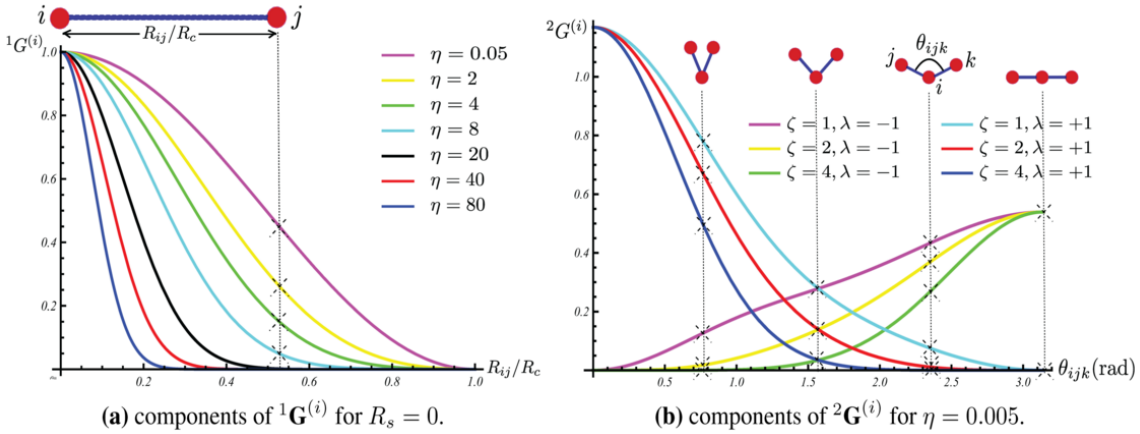


Figure 3: Demonstration of the cutoff radius in a 2D unit cell.



(a) components of $^1\mathbf{G}^{(i)}$ for $R_s = 0$. **(b)** components of $^2\mathbf{G}^{(i)}$ for $\eta = 0.005$.

Figure 4: Visualization of the 1G and 2G Behler descriptors.

- Fewer is better since less variables makes the function smaller, which in turn computes faster.

Finally, for each atom in a system, we calculate the "symmetry function" and pass it to a general feed-forward NN as shown in Figure 5.

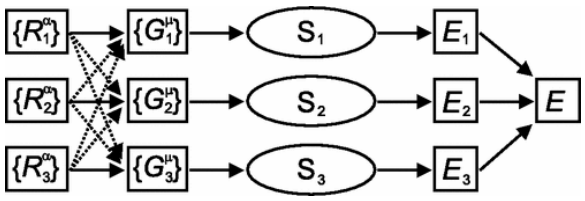


Figure 5: A Behler-Parrinello neural network for a 3 atom system.

- Then we sum the energy contributions from each atom to get the total energy.

- More information can be found in Reference 1.

3 Convergence calculations

First, we need to determine an appropriate level of convergence for our calculations. I usually use the natural bulk configuration of a metal for these studies. For Pd, this is face centered cubic (fcc).

3.1 k-point convergence

First, we determine an appropriate k-point convergence. We will be performing many calculations, so a high level of accuracy is desirable, but not if the computational cost is too high. I use a high energy cutoff (400 eV) to make sure there are no effects from encut convergence to potentially skew the results.

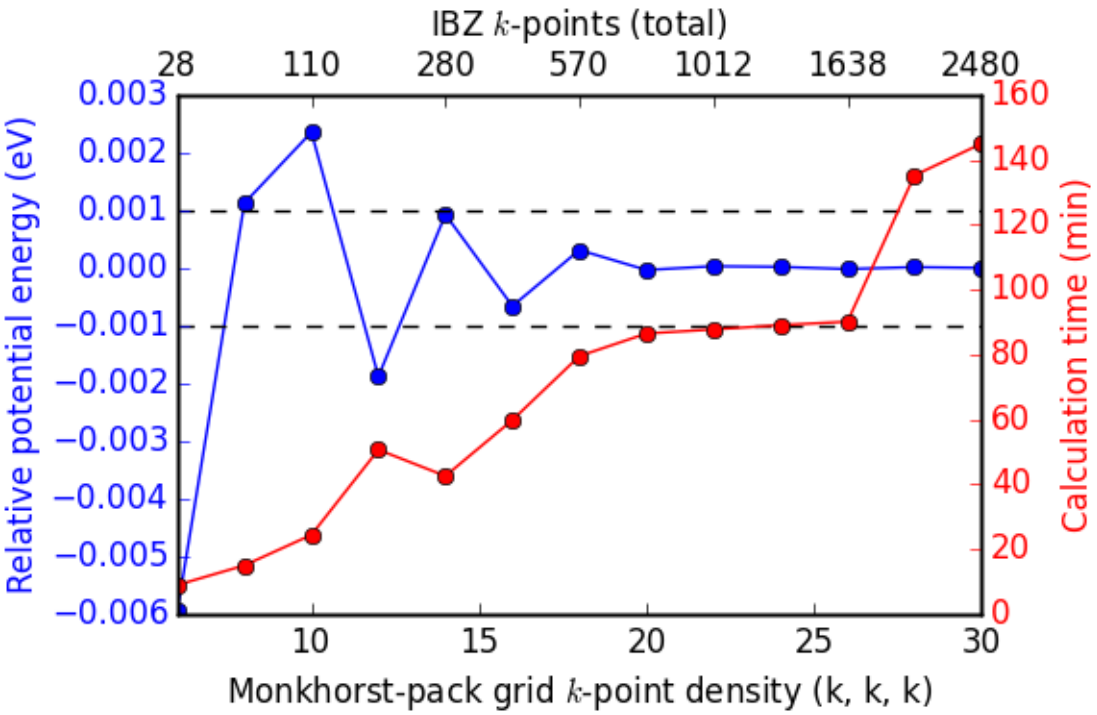


Figure 6: k-point convergence metrics for a single atom unit cell of fcc Pd.

Figure 6 shows that a Monkhorst-pack grid of roughly (14, 14, 14) k-points is sufficient to each 1 meV convergence.

3.2 encut convergence

Next, we look at energy cutoff convergence. Similarly, k-point density is fixed at (16, 16, 16) for these calculations to ensure no effects from lack of convergence.

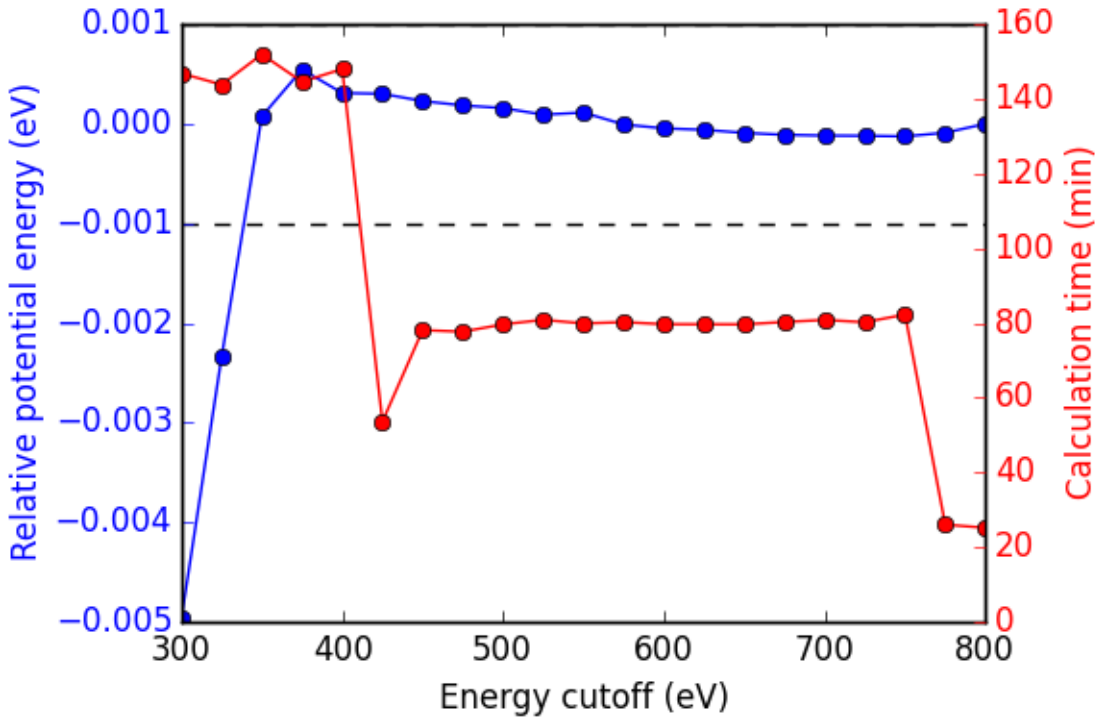


Figure 7: Energy cutoff convergence metrics for a single atom unit cell of fcc Pd.

In this case, Figure 7 shows 350 eV energy cutoff is sufficient to achieve 1 meV convergence. Interestingly, the timing information suggests that 450 eV may be a better choice, but this is difficult to determine with a single run.

3.3 ediff convergence

Finally, we look at the effects of electronic convergence criteria on total energy convergence. For this study, k-points are fixed at (16, 16, 16) and encut at 400 eV.

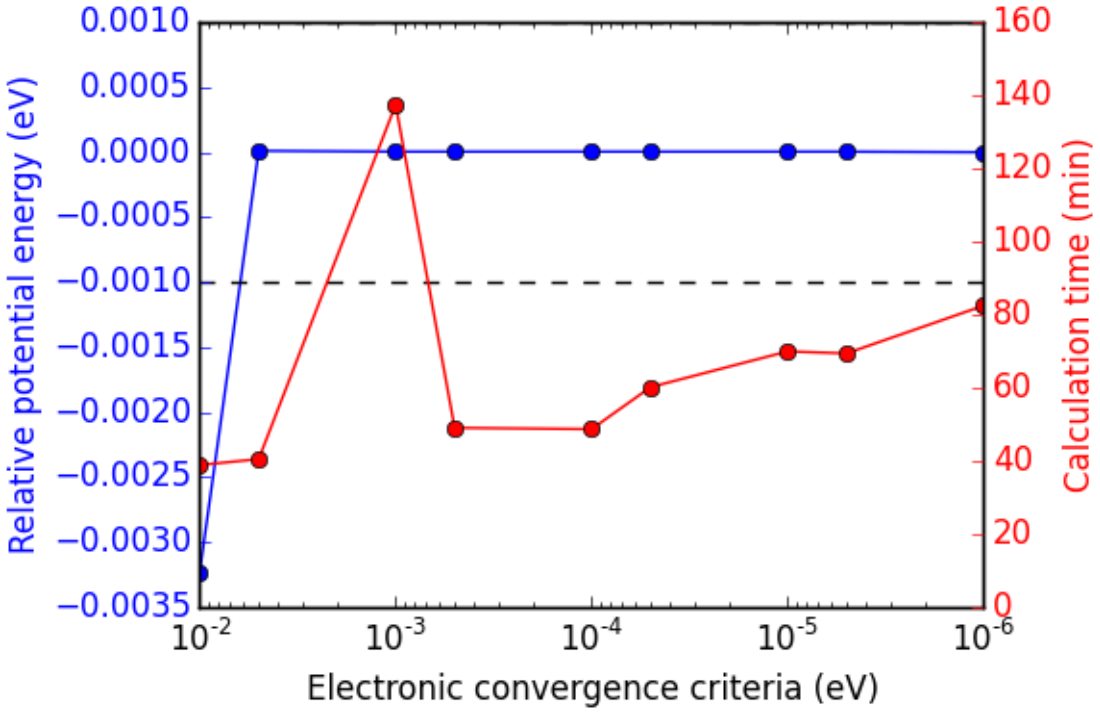


Figure 8: Electronic convergence criteria (ediff) convergence metrics for a single atom unit cell of fcc Pd.

Interestingly, Figure 8 shows that values less than 5e-3 eV (or 5 meV) have no effect on the convergence of the total energy. The calculation times suggest that the default of 1e-4 eV is a good choice.

4 Equation of state

Next we use the convergence criteria to calculate Pd bulk fcc EOS at the desired level of accuracy. I have chosen (14, 14, 14) k-points, 400 eV encut, and 1e-4 eV ediff (default setting). We will need a good sized sample to fit the neural network. I have chosen a fine grid of 71 points about the expected minimum in energy, and 29 additional points to span the space leading to "infinite" separation. Figure 9 shows the resulting fit. The code block also generates an ASE database, which we will use from this point on for easy access to the data. It can be found in the Github repository mentioned in the Introduction.

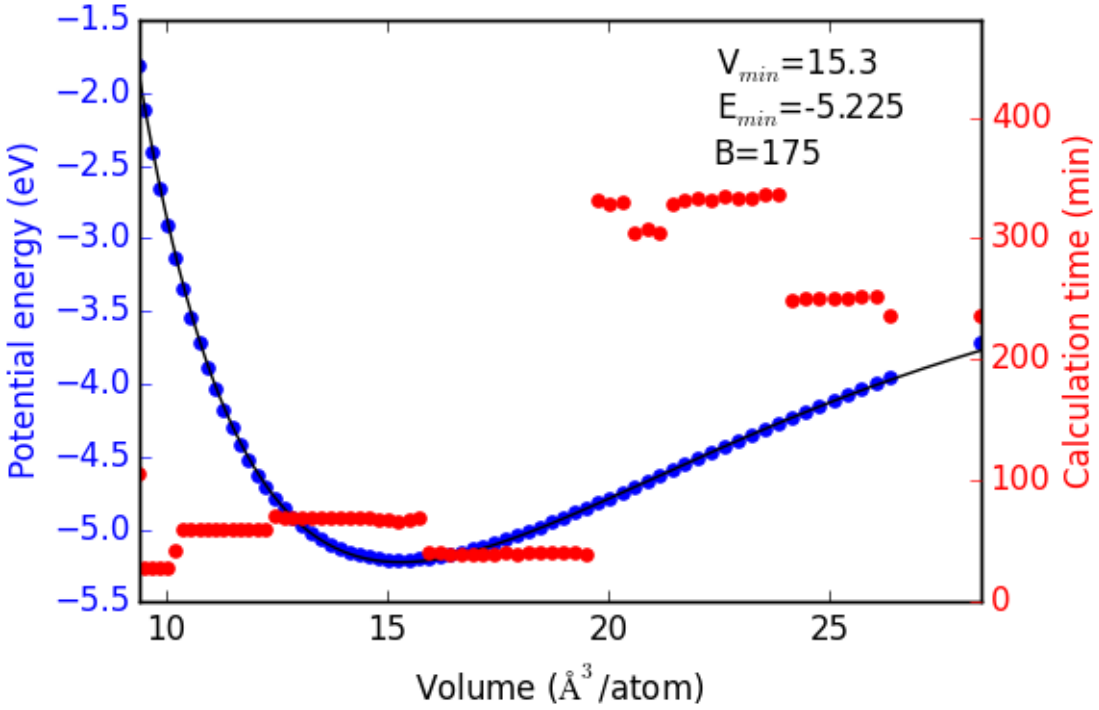


Figure 9: Equation of state for fcc Pd as calculated from DFT.

5 Neural network

To train a neural network we will be using AMP (https://bitbucket.org/andrewpeterson/amp), a software package developed by the Peterson group at Brown University.

Before we begin creating out neural network, we need to separate about 10% of out data into a validation set. This will be useful later, when determining whether over fitting has occurred. There is functionality for this in AMP, but it does not provide with as much control as the following code.

Now we have sudo-randomly labeled 10% of our calculations for validation, and the rest are waiting to be trained in the new train.db file.

5.1 Training neural networks

For all of out neural networks, we will be using the Behler-Parenello (BP) framework for distinguishing between geometries of atoms. Little to no work is published on how to systematically chose an appropriate number of variables for your BP framework, so we simply use the default settings in AMP for now. However, it is worth mentioning that a single G1 type variable (simplest possible descriptor) could be used to describe the fcc EOS, if that is all we are interested in.

We also need to define a cutoff radius for our system which will determine the maximum distance that the BP framework considers atoms to be interacting. 6 Å is a typical value used in the literature for metals with no appreciable long range interactions, which we will be using here.

Finally, it is also often desirable to have multiple neural networks which are trained to the same level of accuracy, but with different frameworks. These frameworks are determined by the number of nodes and hidden layers used. In general, we want the smallest number of nodes and layers possible to avoid the possibility of over fitting. However, too small a framework will be too rigid to properly fit complex potential energy surfaces.

These jobs can be run locally:

We can also submit them to the queue on Gilgamesh:

Once the calculations finish we can check their convergence using the code below. These are trivial networks to train, so convergence should not be an issue. If there is a problem, restart the calculation to try again. This can be a difficult and time consuming part of the process for more complex system.

The single atom unit cell enforces perfect symmetry. This results in cancellation of forces on the atom in the unit cell. Hence, force RMSE = 0.0, which makes for fast training, but less information to train too.

5.2 Validation of the network

Now we need to validate our results to ensure that no over fitting has occurred. First, we will look at the residuals to the training and validation data. Then we will see if the neural networks perform well for their intended purpose. For ease of access, we will add the neural network energy predictions to the database for each structure.

5.2.1 Analysis of residuals

First we look at the residual errors of all the data in the database for each of our frameworks shown in Figure 10. For both fits, the validation set has lower RMSE than the training set. This is a good indication that neither has been over fit, which we can also observe for this simple example, since the validation points follow the same trends observed for the training set data. This is also a good example of how adding additional, unnecessary elements to the framework leads to lower overall fitting accuracy.

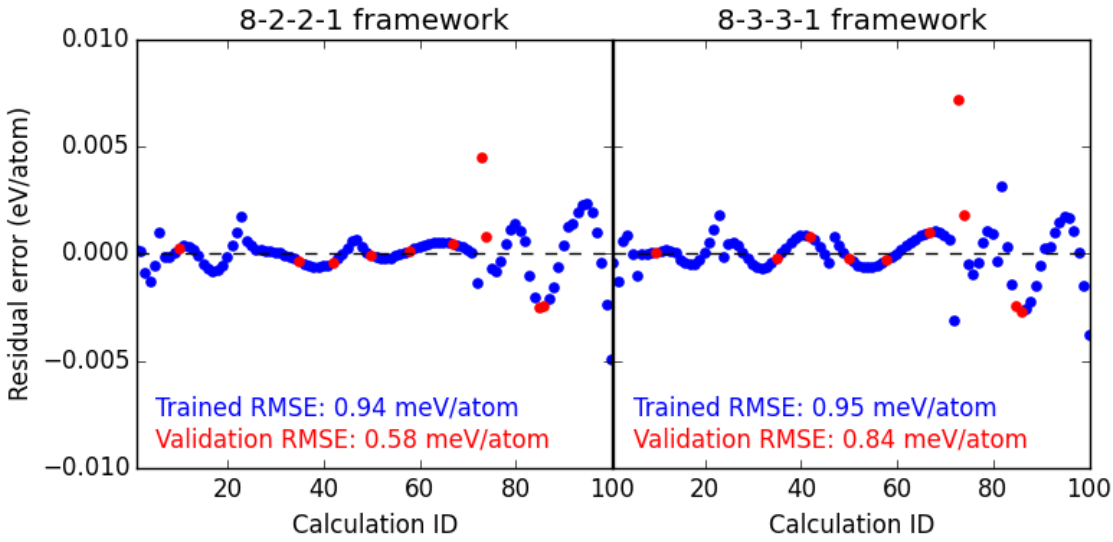


Figure 10: Residual errors to the 2-2 and 3-3 framework neural network.

5.2.2 Recreate the equation of state

Next, we recreate the equation of state using both of the neural networks and the same methodology as with DFT. The results are shown in Figures 11 and 12 for the 2-2 and 3-3 frameworks, respectively.

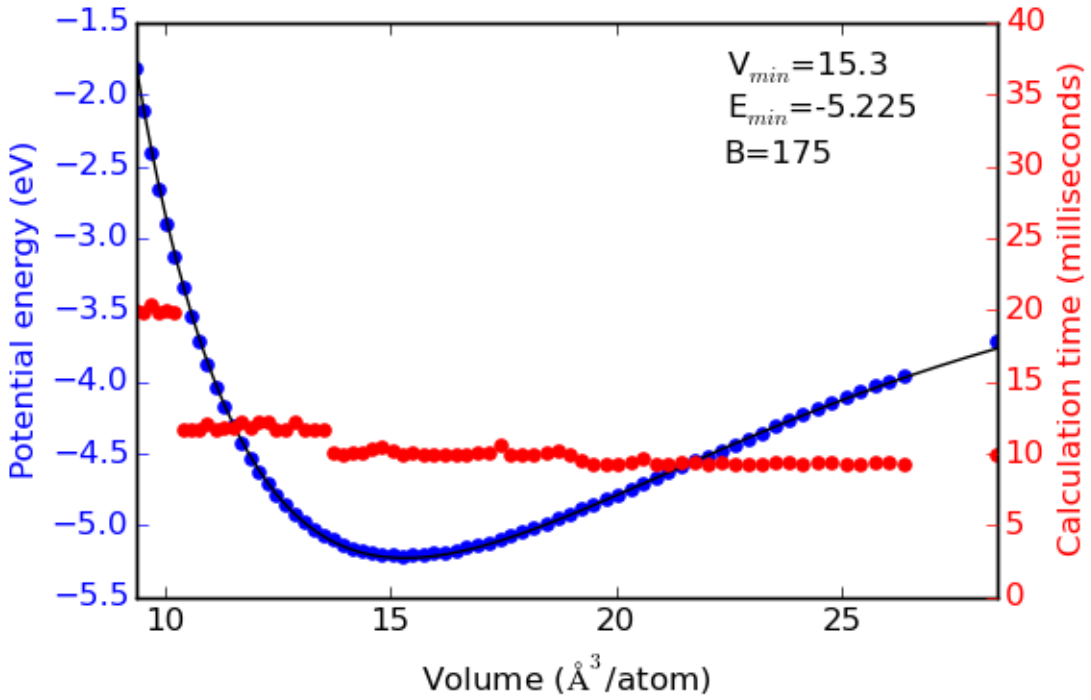


Figure 11: Equation of state for fcc Pd as calculated from a neural network with 2-2 framework.

Each neural network creates an excellent fit to the DFT data, and we see that the calculation speed has improved by up to 6 orders of magnitude in the most extreme cases. For this application the choice of framework seems to have little effect on the equation of state produced.

5.3 Applications

Now we can try and apply our neural networks to things it was not fit to.

For this, we will use or two neural networks jointly which will save us a good amount of

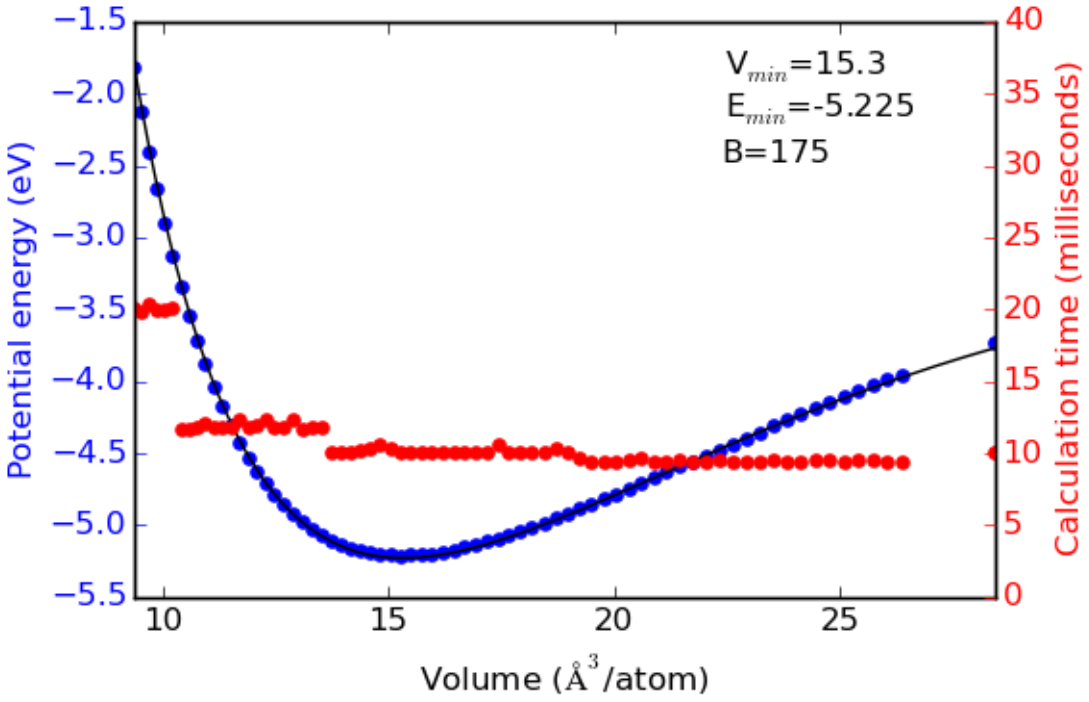


Figure 12: Equation of state for fcc Pd as calculated from a neural network with 3-3 framework.

time validating the networks as we begin to extrapolate. This is demonstrated in the next section.

5.3.1 Geometry optimization

First, we expand the region of equation of state to see how well it extrapolates. In Figure 13, we expand the region of the original equation of state beyond the black dashed lines.

At extreme stretch (factor $> 2.07\%$) both neural networks agree because we have trained it nearly to the cutoff radius of 6.0 Å.

As soon as we strain the lattice below the trained region, the network predictions quickly diverge. This indicates that the training set is not useful for predictions in this region.

We performed 1,000 calculations to produce this figure. To have validated all 1,000 points with DFT would be too time consuming. Instead, we rely on disagreement between neural networks with different framework to probe poorly fitted regions.

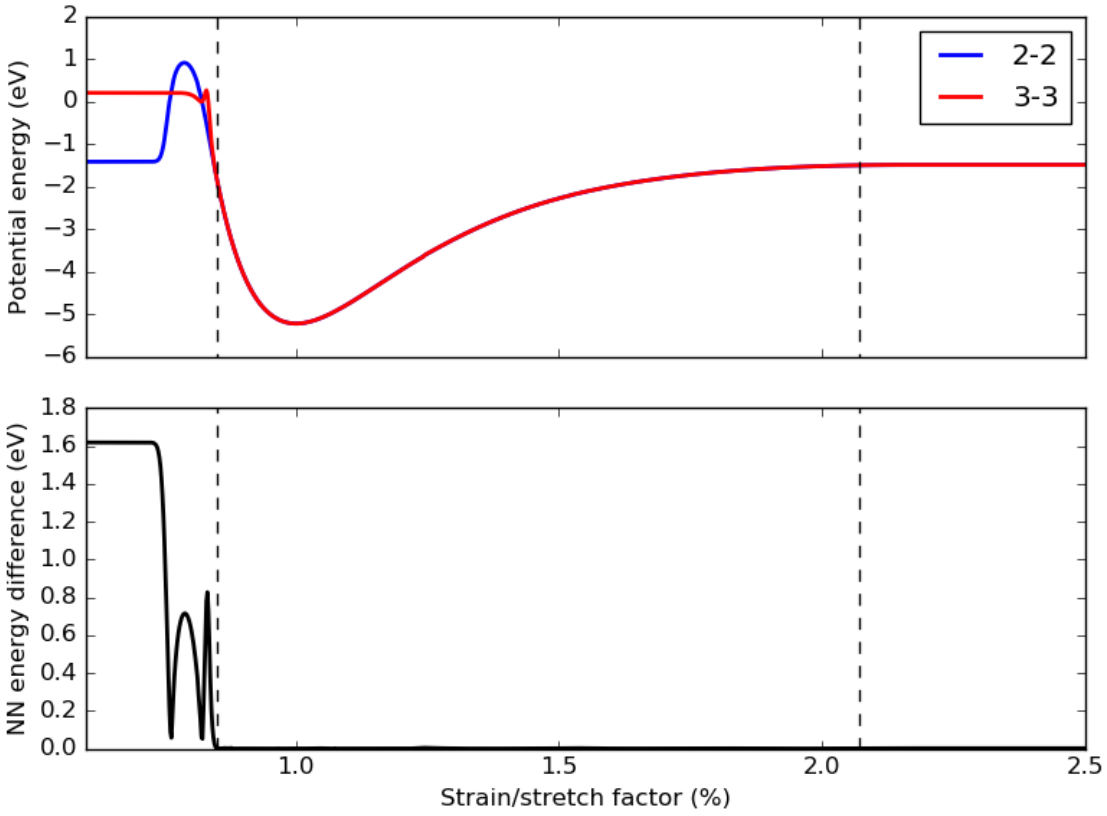


Figure 13: Expansion of the equation of state beyond the region incorporated into the training set.

5.3.2 More complex calculations

Here we attempt to calculate the vacancy formation energy for fcc Pd. This is calculated as shown in Equation 1.

$$E_v = E_f - \frac{n_i - 1}{n_i} E_i \tag{1}$$

from the literature,[2] we know that DFT-GGA should predict a vacancy formation energy of about 1.50 eV.

- Vacancy formation energy with 2-2 framework NN: 4.170 eV

- Vacancy formation energy with 3-3 framework NN: 0.411 eV

neither network does a good job predicting the vacancy formation energy. This is because the networks do not know how to calculate the energy of an fcc lattice with a missing atom.

BFGS: 0 23:22:14 -131.319326 0.6402 BFGS: 1 23:22:19 -131.359079 0.4868 BFGS: 2 23:22:24 -131.407465 0.1290 BFGS: 3 23:22:29 -131.408432 0.1178 BFGS: 4 23:22:34 -131.409158 0.0511 BFGS: 5 23:22:39 -131.406953 0.0411 Vacancy formation energy with 2-2 framework NN: 4.170 eV BFGS: 0 23:22:44 -135.182786 0.0558 BFGS: 1 23:22:49 -135.182910 0.0554 BFGS: 2 23:22:54 -135.183387 0.0320 Vacancy formation energy with 3-3 framework NN: 0.411 eV

5.3.3 Molecular dynamics

Finally, we try an MD simulation. In Figure 14 we begin with a $3 \times 3 \times 3$ primitive unit cell of Pd and add a random amount of kinetic energy to each of the 27 atoms in the system. We then use the forces on those atoms to determine where they will be after a small forward step in time (5 fs). Then, we use the BPNN to calculate the energy and forces on the perturbed system and repeat for 200 time steps.

In Figure 14, the NN energy and corresponding DFT energy of every 4th step is shown. Although the NN predicts the upward trend in energy correctly, the residuals are quite large. This is likely not an acceptable level of error for most applications.

6 Teaching the neural network

6.1 New training set

Here we perform a second iteration of the neural network. Now we will include the DFT validation calculations on the MD simulation shown in Figure 14. The first two sections are repetitions of previous training code shown above.

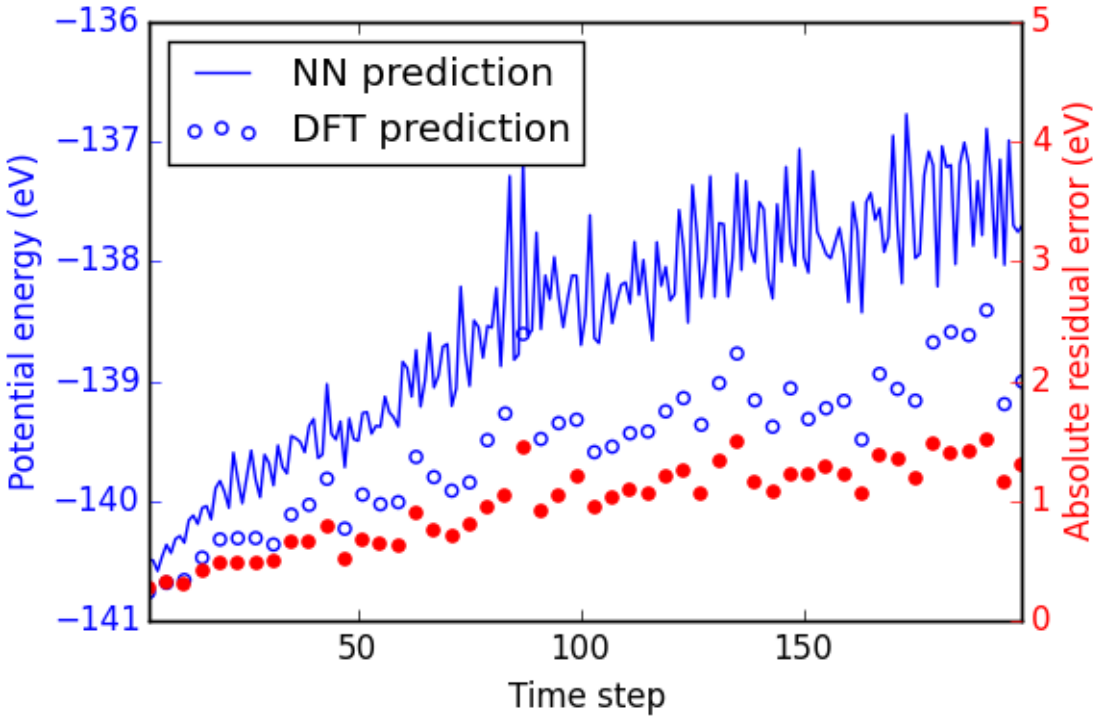


Figure 14: Molecular dynamic simulation of a 3 × 3 × 3 primitive Pd fcc unit cell. First iteration of 2-2 framework predictions.

6.2 Training new network

Training the second networks took significantly longer since we are no longer training such simple structures. There is now a need for more than one descriptor to define the system. However, 8 descriptors is the default for a single element.

6.2.1 Network validation

Since volume is no longer a good description of all structures in our data set, we will simply perform validation of residuals errors based on calculation IDs.

Figure 15 shows that the second iteration of the fit is not as accurate as the first. This is because we have expanded the scope of the potential energy surface we are trying to fit to. Additional accuracy can be obtained by further sampling of similar structures, in this region of the potential energy surface. For most current applications with NN, an RMSE of

≈ 3 meV/atom is considered to be more than sufficient.

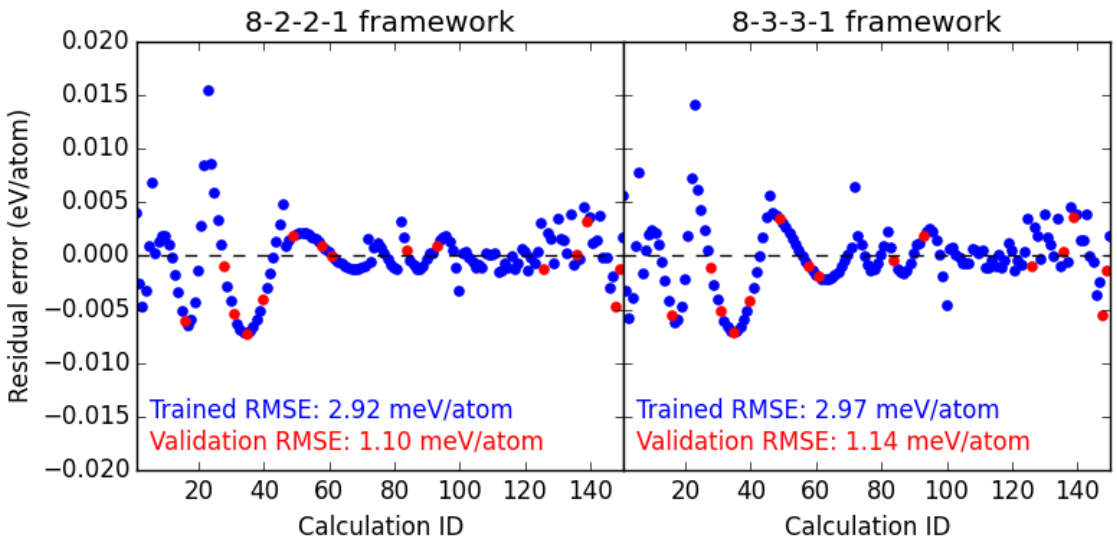


Figure 15: Residual errors to the 2-2 and 3-3 framework neural network after the second iteration of training. Calculation numbers greater than 100 are the MD trajectory structures.

6.3 Attempt 2 with MD simulation

If we re-calculate the energy of the MD trajectory from Figure 14, we can see in Figure 16 that the predictions are greatly improved. Note that the scale of absolute residuals on the right is an order of magnitude lower than before.

Normalizing these residuals on a per atom basis gives absolute residuals errors below 5 meV/atom. This is considered an acceptable level of error for most applications in the literature, but will not be sufficient for all purposes still. Training large systems of atoms to even higher levels of accuracy will become quite difficult since AMP works on a cost function normalized by the number of atoms in each system. This preferentially results in lower levels of absolute residual error for small systems.

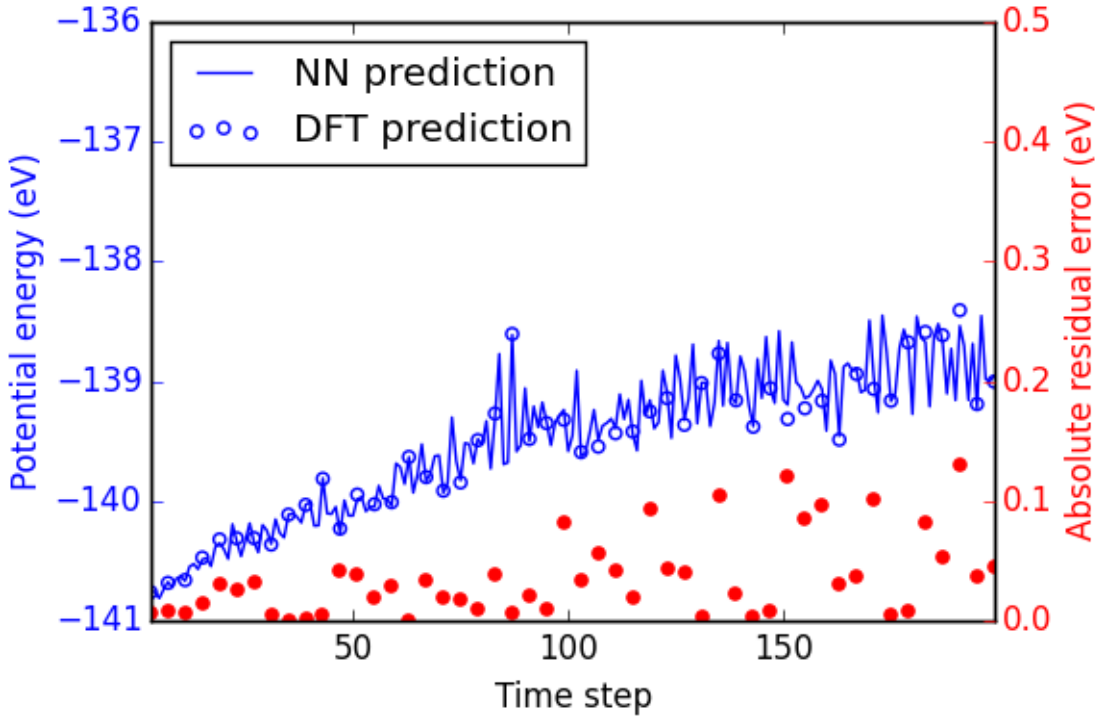


Figure 16: Molecular dynamic simulation of a $3 \times 3 \times 3$ primitive Pd fcc unit cell. Second iteration of 2-2 framework predictions.

References

[1] Jörg Behler, Bernard Delley, Karsten Reuter, and Matthias Scheffler. Nonadiabatic potential-energy surfaces by constrained density-functional theory. *Phys. Rev. B*, 75(11):115409, 2007.

[2] Thomas R. Mattsson and Ann E. Mattsson. Calculating the vacancy formation energy in metals: Pt, Pd, and Mo. *Phys. Rev. B*, 66(21):214110, 2002.