3 APPROACHES TO NEURAL NETWORK POTENTIALS

Xiang Zhang

May 11, 2019

ABSTRACT

A molecule's energy can be calculated from its atoms' positions. Consider the problem of learning this function using neural networks. Several models have been proposed, such as the Behler model, the manybody expansion model, and the Graph Neural Network model. We propose a variant of the manybody expansion model which can also be understood as a generalization of the Behler model. We test the 3 approaches on an overly simple toy dataset. Our proposed model slightly outperforms the other two.

1 Introduction

A molecule is a set of Cartesian coordinates: $\{(x_i, y_i, z_i) : i = 1, 2, \cdots, \text{Natom}\}$. Equivalently, a tensor $\underset{\text{Natom}\times 3}{\mathbf{X}}$ (note that Natom $\times$ 3 describes the shape of $\mathbf{X}$). In this report, we focus on homonuclear gold nanoparticles, and different chemical elements doesn't concern us.

Under the Born-Oppenheimer approximation, the ground state electronic energy of a molecule $\underset{\text{scalar}}{E}$ is a function of $\mathbf{X}$

$$E = E(\mathbf{X})$$

And the force exerted on each atom is also a function of $\mathbf{X}$:

$$\underset{\text{Natom}\times 3}{\mathbf{F}} = -\frac{\partial E}{\partial \mathbf{X}} = \mathbf{F}(\mathbf{X})$$

Our objective is to learn $E(\mathbf{X})$ and $\mathbf{F}(\mathbf{X})$. An estimate of $E(\mathbf{X})$ or $\mathbf{F}(\mathbf{X})$ is called a force field or interatomic potential, and the search for it, whether using neural networks or not, is a long-standing[1] and active field of research for quantum chemists and computational materials scientists.

$E(\mathbf{X})$ is completely defined by the Schrodinger equation, and there are many ways to partially or approximately solve the equation. Almost accurate calculations of E using, say, Quantum Monte Carlo[2], is too computationally expensive due to manybody interactions; anything larger than a water molecule is hardly tractable. A less accurate approximation, Density Functional Theory (DFT) [3] is very popular[4] among material scientists. One calculation of $E_{\text{DFT}}(\mathbf{X})$ typically takes hours to days. People would calculate $E_{\text{DFT}}(\mathbf{X})$ on tens of thousands of molecules and use the data to train neural networks. Far less accurate (and special purpose only) approximations based on analytic, closed-form expressions such as the Effective Medium Potential (EMT)[5] also exist. In this report, we use the EMT potential to generate training data.

The costly DFT calculations motivate the use of neural networks to learn $E_{\text{DFT}}(\mathbf{X})$ or any accurate $E(\mathbf{X})$. The notion of using neural network is not new[6]. I have not been able to find a review article on all the neural network potential models proposed so far, but here are the three relevant approaches that I came across. The Behler approach[7] computes the fingerprint of the local environment of each atom using empirical functions, passes the atom-wise fingerprints through MLPs, and sums the output over all atoms. The Graph Neural Network approach[8] represents the molecule as a graph, and estimates E using Graph Neural Networks such as Graph Convolutional Networks[9] and Gated Graph Sequence Neural Networks[10]. A seemingly less popular approach [11] builds on the idea manybody expansion: E is contributed by interactions between atom pairs, triplets, quadruplets, etc, and the lower-order manybody expansion terms are more important.

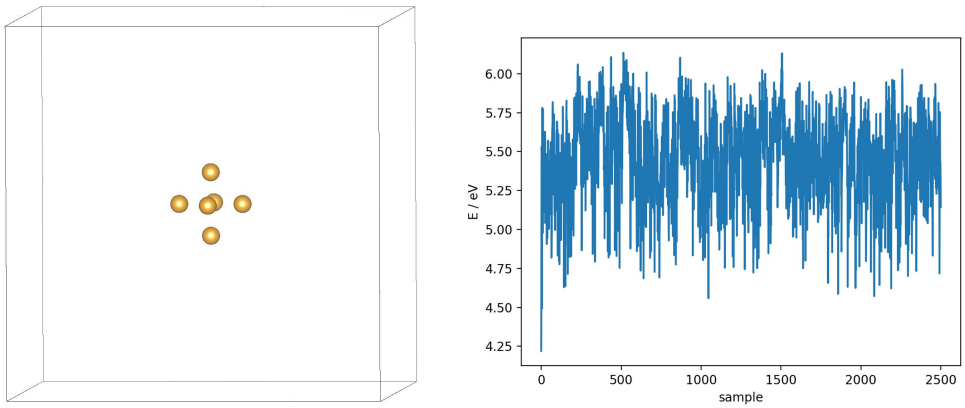
In this report, we test the 3 aforementioned approaches to constructing neural network potentials on a Au molecule dataset calculated using EMT approximation. Using the Behler method as-is, we achieve good performance on $E(\mathbf{X})$

and bad on $\mathbf{F}(\mathbf{X})$. Using a slightly modified version of the manybody expansion approach (to be honest, I did not read through all the manybody expansion papers), we achieve very good performance on $E(\mathbf{X})$. Using a very simple Graph Neural Network, we achieve a mediocre performance on $E(\mathbf{X})$.

2 Data

In section, we describe how the dataset is collected.

The overall idea is to use EMT potentials to calculate $E(\mathbf{X})$ on a large number of molecules to form a dataset (Figure 1), and train machine learning models on it. Each sample in the dataset is one molecule, with feature $\underset{\text{Natom}\times 3}{\mathbf{X}}$ and label $\underset{\text{scalar}}{E}$ .The coordinate of atom i is denoted by $\underset{3}{\overrightarrow{x_i}}$. The distance between atoms i and j is denoted by $r_{ij} = |\overrightarrow{x_i} - \overrightarrow{x_j}|$.



a) Molecule 1 b) All E's

Figure 1: Data

The dataset is created using molecular dynamics as follows. "Carve out" 6 Au atoms from the crystalline Au structure as the starting structure $\mathbf{X}_1$. Calculate its energy E_1 using the EMT potential as implemented in the ASE/Asap[12] code. Assign all the atoms an initial velocity according to the Maxwell-Boltzmann velocity distribution corresponding to 300K. Run NVE molecular dynamics (as implemented in ASE) for 125,000 steps with a time step of 1fs. Record X and E every 50 timesteps to form the dataset. As far as I am aware of, all the parameters are ones commonly used in publications.

Our dataset consists of Au_6 molecules calculated using the cheap EMT potentials. It is a toy dataset. As a reference, Ref. [13] studied Au_{25-60} molecules (much larger) using DFT potentials (much, much more accurate and complicated). We use a smaller molecule and a much simpler potential, because this way we won't have to worry about working with limited data, and errors when learning a transparent model are easy to analyze. We had plans for moving on to more complicated datasets, but we did not even have enough time to finish working on this toy dataset.

2.1 Intuitions

Chemistry is about developing intuitions (covalent, metallic and ionic bonds) about $E(\mathbf{X})$ for certain $\mathbf{X}$'s (molecules, crystals, polymers). Pre-machine-learning force fields do the same. We can gain some intuition by inspecting them.

The AMBER force field[14] is one of the more popular "far less accurate" force fields. The main component of its $E(\mathbf{X})$ is

$$E_{\text{AMBER}} = \sum_{\text{bonds}} K_r(r_{ij} - R_0)^2 + \sum_{\text{angles}} K_\theta(\theta_{ijk} - \theta_0)^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}\left[1 + \cos(n\phi_{ijkl} - \phi_0)\right] + \sum_{i<j}\left[\frac{A_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\epsilon r_{ij}}\right]$$

where $\theta_{jik} = \langle \overrightarrow{x_j} - \overrightarrow{x_i}, \overrightarrow{x_k} - \overrightarrow{x_i} \rangle$, $\phi_{ijkl} = \langle \hat{n}_{ijk}, \hat{n}_{jkl} \rangle$.

The most obvious trait of $E(\mathbf{X})$ is that E is a scalar for any shape of $\mathbf{X}$ (any number of atoms). All MLP-based models must be able to convert a non-fixed-shape input into a fixed-shape input for use by neural networks.

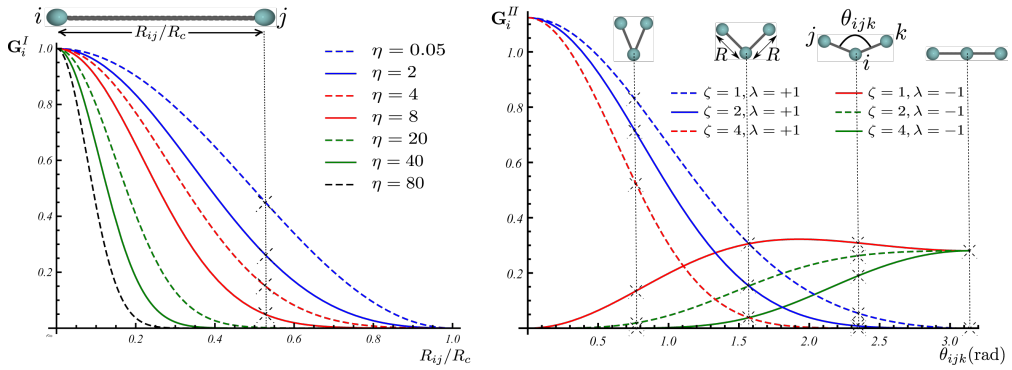


Figure 2: Gaussian fingerprints. Source: AMP website

As E_{AMBER} depends on scalars such as spatial distance r_{ij} and angles between two vectors θ_{ijk} or planes ϕ_{ijkl} and not directly on vectors $\overrightarrow{x_i}$, it is independent of the coordinate system, and translationally, rotationally and permutationally invariant. Obviously, this is true not only for $E_{\text{AMBER}}(\mathbf{X})$, but for accurate and almost all approximate $E(\mathbf{X})$'s. Accordingly, most neural network models proposed so far are subject to the same symmetry.

E_{AMBER} is local: loosely, $\lim_{r_{ij}\to\infty} E = 0$. This is true for accurate and all approximate $E(\mathbf{X})$'s. Accordingly, neural network models often apply a cutoff range, beyond which two atoms do not interact.

E_{AMBER} is a sum over 1-body terms (chemically, free atom energies), 2-body terms (stretching of bonds, Coulomb interaction, van der Waals interaction), 3-body terms (angular), and 4-body terms (torsion). The chemical intuition is that the permutationally invariant $E(\mathbf{X})$ can be expressed in such a "manybody expansion" format, where lower-order terms are assumed to hold the most significance. I do not know of a rigorous proof of this and doubt if one exists.

3 Behler model

The Behler-Parrinello model is[7]

$$
\begin{aligned}
E &= \sum_i \text{MLP}\left[\begin{array}{l} G_i^{\text{rad}} = \sum_{j\neq i} e^{-\eta r_{ij}^2} f_c(r_{ij}) \\ G_i^{\text{angle}} = 2^{1-\zeta} \sum_{j,k\neq i} (1+\lambda\cos\theta_{jik})^\zeta e^{-\eta(r_{ij}^2+r_{jk}^2+r_{ik}^2)} \cdot f_c(r_{ij}) f_c(r_{jk}) f_c(r_{ik}) \end{array}\right] \\
f_c(r) &= 0.5\left(1+\cos\pi\frac{r}{R_c}\right) I(r<R_c) \qquad (1) \\
\mathbf{F} &= -\frac{\partial E}{\partial \mathbf{X}}
\end{aligned}
$$

The MLP does not have 2 input nodes, but 4, 6, 8 or more: there are multiple (typically 10-20 combinations of) $\zeta, \lambda, \eta, R_C$'s. They are not trainable parameters and are decided by the user based on intuition. When multiple elements are present, each atom pair (or triplet) are summed separately and all fed into one MLP.

3.1 Intuition

The central idea of the Behler model is to fingerprint atom i's local environment, pass it through an MLP to get atom i's energy E_i, and sum over i.

The fingerprinting process reduces an arbitrarily shaped tensor to a fixed shape. It is therefore important that this compression loses as little relevant information as possible. The intuition of the Gaussian fingerprints is shown in Figure 2. Intuitively, G_i^{rad} terms characterize 2-body interactions by aggregating r_{ij} over all neighboring atom j's; equivalently, we can say it is trying to count how many atom j's are at a distance of R_C away from atom i, forming a shell-like structure. Similarly, G_i^{angle} terms try to capture specific chemically relevant triplet configurations, say, 3 atoms that form a H_2O molecule. f_c serves as a cutoff.

3.2 Implementation details

The Atomistic Machine-learning Package[15] fully implements the Behler model with Gaussian fingerprints, including preprocessing. It also provides a sensible default set of $\zeta, \lambda, \eta, R_C$'s; we use the default. $\mathbf{F}$ is calculated by manually calculating the analytic derivates instead of backprop all the way through the fingerprints (kudos to them), and the entire package builds on numpy and not tensorflow etc. MLPs are trained using scipy's optimization algorithms, and terminate automatically.

As correlation exists between samples that have similar indices (see appendix A), we do not shuffle samples. If we have 2500 samples, we use 1-2000 as training set and 2000-2500 as test set. Due to the correlation, we are unabe to properly cross-validate.

3.3 Results: Au_6 dataset

Using the Behler model, we trained $E(\mathbf{X})$ on the aforementioned Au_6 dataset. We tried different MLP shapes (number of layers and number of nodes per layer). The results are shown in Table 1.

Table 1: Training and test losses of Behler model on Au_6 dataset

MLP architecture	RMSE (training) / eV	RMSE (test) / eV
(5, 5)	0.08	0.09
(10, 10, 10)	0.04	0.04
(50, 25, 10, 10)	0.01	0.30

The $(5, 5)$ model is underfitting. The $(50, 25, 10, 10)$ is too expressive and overfitting. The (relatively) optimal MLP has hidden layers of shape $(10, 10, 10)$ (input and output layers omitted). It achieves an RMSE of 0.04eV. The predicted and actual values of E in training and test sets are plotted in Figure 3. There are no outliers or apparent systematic errors.

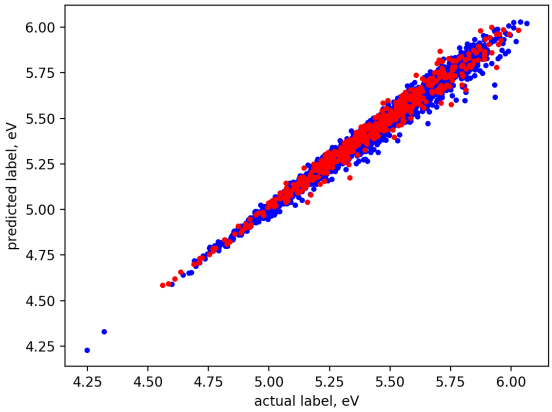


Figure 3: $E_{\text{predicted}}$ vs E_{actual} for training (blue) and test (red) set, Behler model

3.4 Results: Pt_8Au_2 dataset

We also applied the Behler model to a slightly more complicated system: a periodic system where Au adsorbs onto the 110 surface of fcc Pt crystals. We trained not only $E(\mathbf{X})$ but also $\mathbf{F}(\mathbf{X})$, as the Behler model allowed training of both. The hyperparameters we tuned include dataset size and MLP architecture. Training results are listed in Tables 2 and 3.

In Table 2, lines 1 and 3 are overfitting, and are solved by increasing the number of samples. Line 2 is underfitting, and is solved by increasing the number of layers.

In Table 3, line 2, we achieve good performance on $E(\mathbf{X})$. However, $\mathbf{F} = -\frac{\partial E}{\partial \mathbf{X}}$ is consistently strongly overfitting. It may be due to the fact that we're training E and $\mathbf{F}$ simultaneously and penalizing $\mathbf{F}$ losses too much. It is also possible

Table 2: Training and test losses of Behler model on Pt_8Au_2 dataset; training E only

Dataset size	MLP architecture	RMSE (training) / eV	RMSE (test) / eV
250	(10,10)	0.0001	0.42
2500	(10,10)	0.03	0.05
2500	(10,10,10)	0.007	0.13
25000	(10,10,10)	0.026	0.028

Table 3: Training and test losses of Behler model on Pt_8Au_2 dataset; training E and F simultaneously

Dataset size	MLP architecture	E		F	
		RMSE (training) / eV	RMSE (test) / eV	RMSE (training) / eV/A	RMSE (test) / eV/A
250	(5,5)	0.017	0.19	0.26	45.3
2500	(10,10,10)	0.08	0.07	0.3	14.3

that $\mathbf{F}$ is inherently more complicated than E (having 3 components, thus a direction and much less symmetry) and is harder to train properly. Another explanation of $\mathbf{F}$ overfitting is as follows. Because $\mathbf{F}$ is the derivative of E, knowing $\mathbf{F}$ at $\mathbf{X} = 1$ (ignore the shapes for now) gives us information about E at $0.98, 0.99, 1, 1.01, 1.02$. Having multiple $\mathbf{F}$'s means having multiple clusters of $E(X)$'s. This is not ideal; we want $\mathbf{X}$ to be spaced equally rather than clustered together, and in this sense, our training data for $\mathbf{F}(\mathbf{X})$ is simultaneously evenly spaced and not evenly spaced. It feels slightly far-fetched, though.

We did not have time to apply other models to the $\mathrm{Pt_8Au_2}$ dataset. That would require i) properly encoding element types (possibly one-hot, or one MLP per element type), and ii) taking into consideration interactions between atoms in neighboring mirror periodic units.

4 Manybody expansion model, $\mathbf{E}(\mathbf{X})$

4.1 Intuition: inspecting EMT's $\mathbf{E}(\mathbf{X})$

As we're using a toy function to generate our dataset $E(\mathbf{X})$, it is possible to dive into the source code and see the "answer" of the regression problem. Inspecting the source code reveals that

$$\begin{aligned}\sigma_i &= \sum_j f(r_{ij}) \\ E_i &= g(\sigma_i) \\ E_{ij} &= h(r_{ij}) \\ E(\mathbf{X}) &= \sum_i E_i + \sum_{ij} E_{ij}\end{aligned}$$

where f, g, h are normal univariate functions. Note that we'll reuse f, g, h, but each time they will refer not to the same function, but merely the fact that it is a univariate function.

As MLPs are capable of approximating any fixed-size functions on $[-1, 1]$ to arbitrary accuracy, we can simply replace f, g, h each with a different MLP, and obtain a hypothesis that is guaranteed to solve the problem:

$$E = \sum_i \mathrm{MLP}\left(\sum_j \mathrm{MLP}\left(r_{ij}\right)\right) + \sum_i \mathrm{MLP}\left(\sum_j \mathrm{MLP}\left(r_{ij}\right)\right) \tag{2}$$

4.2 Intuition: extending the manybody expansion model

One of the intuitions of the AMBER force field is that E can be decomposed into a sum of 1-body, 2-body, 3-body… terms:

$$E = \sum_i E_i + \sum_{ij} f(r_{ij}) + \sum_{ijk} f(r_{ij}, r_{jk}, r_{ik}) + \cdots$$

A few models have been proposed based on this formulation[11].

Note that symmetry requires that 1-body terms only depend on element types. In our case, there are no 1-body terms. We also know from section 4.1 that there are no 3-body terms. Thus the standard manybody expansion model is reduced to

$$E = \sum_{ij} \text{MLP}(r_{ij}) \tag{3}$$

Note that we do not include 3-body terms here. As can be seen in Section 4.1, our $E(\mathbf{X})$ should only include up to 2-body terms.

We propose to couple 2-body terms into 1-body terms:

$$E = \sum_{i} \text{MLP}\left(\sum_{j} \text{MLP}\left(r_{ij}\right)\right)$$

Which is the same as Eq. 2 in section 4.1. Intuitively, the new 1-body term E_i depends not only on atom i, but its environment: a sum over all of its neighbors j. Mathematically, it is a crossover between 2-body terms and 1-body terms beyond the expressive power of standard manybody expansion.

Naturally, the next question is whether we can couple 3-body terms into lower-order terms:

$$E = \sum_{i} \text{MLP}\left(\sum_{j} \text{MLP}\left(\sum_{k} \text{MLP}\left(r_{ij}, r_{ik}, r_{jk}\right)\right)\right) \tag{4}$$

Wrapping an MLP outside the outer sum may provide extra predictiveness:

$$E = \text{MLP}\left(\sum_{i} \text{MLP}\left(\sum_{j} \text{MLP}\left(\frac{1}{r_{ij}}\right)\right)\right) \tag{5}$$

4.3 Intuition: generalizing the Behler model

The empirical forms of G_i^{rad} and G_i^{angle} in the Behler model

$$E = \sum_{i} \text{MLP}\left[\begin{array}{l} G_i^{\text{rad}} = \sum\limits_{j \neq i} e^{-\eta r_{ij}^2} f_c(r_{ij}) \\ G_i^{\text{angle}} = 2^{1-\zeta} \sum\limits_{j,k \neq i} (1 + \lambda \cos\theta_{jik})^{\zeta} e^{-\eta(r_{ij}^2 + r_{jk}^2 + r_{ik}^2)} \cdot f_c(r_{ij}) f_c(r_{jk}) f_c(r_{ik}) \end{array}\right]$$

is rather weakly justified. If we replace those functions with MLPs

$$E = \sum_{i} \text{MLP}\left[\begin{array}{l} \sum\limits_{j} \text{MLP}(r_{ij}) \\ \sum\limits_{jk} \text{MLP}(r_{ij}, r_{ik}, r_{jk}) \end{array}\right]$$

again we arrive at Eq. 2 and Eq. 5.

4.4 Implementation details

Take Eq. 2 for example:

$$E = \sum_{ij} \text{MLP}(r_{ij})$$

It can be converted to tensor form:

$$\underset{\text{Nsample}}{\mathbf{E}} = \sum_{\text{Nsample}} \underset{\text{Nsample}\times\text{Natom}\times\text{Natom}}{\text{MLP}}\left(\underset{\text{Nsample}\times\text{Natom}\times\text{Natom}}{\mathbf{R}}\right)$$

This can in turn be implemented using sequential or functional Keras[16] models. In particular, MLP is implemented as a few consecutive Dense layers, and $\sum$ is a `K.sum` lambda layer.

The exact choice and standardization of features is included in appendix B

4.5 Results and analysis

Training and test RMSE for Eq. 2-5 are shown in Table 4.

Table 4: Performance of manybody expansion models on $E(\mathbf{X})$ data

Mode	MLP architecture	Epochs	ϵ_{train}	ϵ_{test}
Eq. 3	(20,50,1),(1,50,20)	200	0.1	0.1
Eq. 2	(20,50,1),(1,50,20)	200	0.017	0.014
Eq. 4	(10,50,1),(10,50,10),3,50,10)	400	0.013	0.009
Eq. 5	(20,50,1),(20,50,20),(1,50,20)	200	0.014	0.012

The classical manybody expansion (Eq. 3) fails to deliver. It is clearly underfitting and the model is not expressive enough.

Coupling 2-body terms into 1-body terms (2) provides the necessary expressiveness, and produces near-perfect performance, outperforming both the Behler and the Graph Neural Network models using less time. However, I must point out that "cheating" (section 4.1) is how I actually arrived at Eq. 2. As such, Eq. 2's superior performance does not strongly indicate that it is the superior model; it was engineered to work with this particular EMT dataset after all.

The extra expressiveness achieved by wrapping an MLP over the outmost sum provides no visible improvement.

Apparently, coupling 3-body terms into lower-order terms (Eq. 4) achieves as much, if not higher accuracy, indicating that higher order coupled terms "include" lower order coupled terms. It also proves that the trick of coupling higher order terms into lower order terms is generalizable. Also, 2-body terms coupled into 1-body terms (Eq. 2) "include" pure 2-body terms (Eq. 3).

4.6 Next steps

There are a few ways in which our toy dataset differs from, and our model can fail on, realistic ones.

Firstly, Au_6 is a small system. One known problem with the manybody expansion approach is the number of terms in the sum[11]. Imagine a quantum dot of 100 atoms. The triplet term sums over $C^3_{100} = 161700$ MLP outputs; it may lead to exploding gradients. One possible solution is to average rather than sum, but that brings scaling issues.

Secondly, we did not deal with different types of elements. Taking it into consideration, a molecule is actually defined by $\{(x_i, y_i, z_i, s_i) : i = 1, 2, \cdots, \text{Natom}\}$. As Professor Jegelka pointed out, type s_i can either by one-hot encoded if interactions between different types of elements have a shared componenet, or each type can have its own MLP if different elements behave very differently.

Thirdly, the EMT potential is an analytical, closed-form one. Our toy dataset is a little too easy. There is also absolutely no reason why the "cheated" models (Eq. 2) will perform as well on non-EMT datasets. Although, since the EMT potential is a good approximation to actual potentials, we wouldn't expect the "cheated" models to completely fail either.

Lastly, as pointed out by Hongyin, it's rather interesting that MLPs are able to learn Behler's empirical G functions. What did the MLPs learn?

5 Manybody expansion model, $\mathbf{F}(\mathbf{X})$

5.1 Intuition

The manybody expansion formulation naturally allows the learning of $\mathbf{F} = -\frac{\partial E}{\partial \mathbf{X}}$. This is not a trivial sub-problem having solved $E(\mathbf{X})$. Many physical observables correspond to 1 vector per atom; solving $\mathbf{F}(\mathbf{X})$ can potentially solve those questions as well. For example, I have been working on geometry optimizations, where $E(\mathbf{X})$ is minimized with respect to $\mathbf{X}$ starting from $\mathbf{X}_0$, and $\mathbf{X}^*(\mathbf{X}_0)$ is a desirable objective function.

There are a few intuitive formulations of $\mathbf{F}(\mathbf{X})$.

The intuition of the mixed-1-2-body-terms is that atom i's contribution to total energy is affected by its environment (a sum of over its neighbors). Thus, $\overrightarrow{f_i} = -\frac{\partial E}{\partial \overrightarrow{x_i}}$ should also be decided by its environment:

$$\overrightarrow{f_i} = \text{MLP}\left(\sum_j \text{MLP}\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right) \tag{6}$$

Note that rotational symmetry no longer applies, but translational symmetry still does.

Alternatively, by cheating, we know that

$$E = \sum_i f\left(\sum_j g\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right)$$

and because physics says $\overrightarrow{f_k} = -\frac{\partial E}{\partial \overrightarrow{x_k}}$, we can take the derivative

$$\begin{aligned}\overrightarrow{f_k} &= -\frac{\partial}{\partial \overrightarrow{x_k}}\left(\sum_i f\left(\sum_j g\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right)\right) \\ &= \sum_i f\left(\sum_j \left(g\left(\overrightarrow{x_i} - \overrightarrow{x_k}, \overrightarrow{x_j} - \overrightarrow{x_k}\right)\right)\right) h(\overrightarrow{x_i} - \overrightarrow{x_k}) + u\left(\sum_i v(\overrightarrow{x_i} - \overrightarrow{x_k})\right)\end{aligned}$$

Replacing f, g, h with MLP's and removing unnecessary terms, we get

$$\overrightarrow{f_i} = \sum_i \text{MLP}\left(\sum_j \text{MLP}\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right) \tag{7}$$

or, a bit more expressive (in case my math is wrong)

$$\overrightarrow{f_i} = \text{MLP}\left(\sum_i \text{MLP}\left(\overrightarrow{x_i} - \overrightarrow{x_k}, \sum_j \text{MLP}\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right)\right) \tag{8}$$

Yet another way is to directly cheat; that is, to read the source code to see how it calculates f_i, thus obtaining a fool-proof fail-safe solution:

$$\overrightarrow{f_i} = \sum_i \text{MLP}\left(\overrightarrow{x_i} - \overrightarrow{x_k}, \text{MLP}\left(\sum_j \text{MLP}\left(\overrightarrow{x_j} - \overrightarrow{x_i}\right)\right)\right) \tag{9}$$

5.2 Results and analysis

Training and test losses using each of the aforementioned model are listed in Table 5.

Table 5: Performance of manybody expansion models on $\mathbf{F}(\mathbf{X})$ data

Model	MLP architecture	Epochs	RMSE (train) / eV/Å	RMSE (test) / eV/Å
Eq. 6	(20,3),(20,20),(20,20)	200	0.69	0.7
Eq. 7	(20,3),(20,20),(20,20)	200	0.27	0.4
Eq. 7	(20,3),(20,20),(20,20)	1000	0.27	0.31
Eq. 8	(50,3),(50,20),(50,20)	1200	0.11	0.16
Eq. 9	(50,3),(50,20,20),(50,20,20)	9000	0.08	0.12

Firstly, we are unable to achieve a satisfactory loss on the $\mathbf{F}(\mathbf{X})$ dataset. As the dataset is generated using a toy model and we are "cheating", an almost perfect accuracy is to be expected, while the best achieved is 0.12, using the "cheating" model Eq. 9.

Secondly, on the same dataset (same $\mathbf{X}$), training $\mathbf{F}$ requires much longer training time (hours instead of minutes) compared to $\mathbf{E}$, and produce much poorer results, overfitting slightly all the way. It is curious considering that the Behler model is also unable to satisfactorily learn $\mathbf{F}$. It is possible that $\mathbf{F}$ is simply difficult to learn (see section 3.4 for more discussion).

6 Graph Neural Networks

A molecule is represented as an undirected complete graph (V, E), where node i contains $\overrightarrow{x_i}$.

In the message passing phase,

$$m_v^{t+1} = \sum_{w \in N(v)} \text{concat}(h_v^t, h_w^t)$$
$$h_v^{t+1} = \text{Dense}(m_v^{t+1}) \tag{10}$$

where $N(v)$ denotes all neighbors of node v.

In the readout phase,

$$E = \sum_v \text{MLP}(h_v) \tag{11}$$

The algorithm is implemented using the DGL library. With 2000 training samples, message passing Dense layers $(6, 12), (24, 6)$, readout layers $(6, 12, 1)$, we achieve an RMSE of 0.2eV, lower than both the Behler model and our manybody expansion models.

Theoretically, graph neural networks is the most representative of the three. In particular, manybody expansion models can be considered 1 or 2 message passing iterations. This Graph Neural Network model probably is not expressive enough; unfortunately, we did not have time to try another one.

7 Summary

We have tested 3 classes of models on a Au_6-EMT dataset: the Behler-Parrinello model, 4 variations of manybody expansion model, and a Graph Neural Network model. Their losses on $\mathbf{E}(\mathbf{X})$ are compared in Figure 4. As we're using a toy dataset, there is no commonly recognized baseline; the RMSE of the Behler model on a DFT Au_{50} dataset[13] is included for reference.

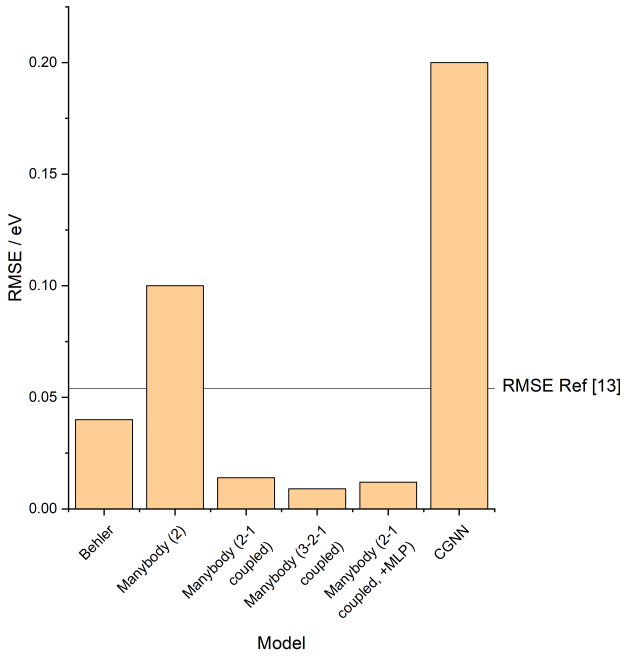


Figure 4: $\mathbf{E}(\mathbf{X})$ RMSE of all tested models

As is to be expected, the well-established Behler model performs reasonably well. We could not find a good Graph Neural Network model due to a lack of time. Of the 3 classes, the manybody expansion model, and in particular Eq. 4,

where we proposed to couple 3-body terms into lower-order terms, has the best performance (RMSE<0.01eV). None of the models perform too well when it comes to learning $\mathbf{F}(\mathbf{X})$, however, with a best RMSE of 0.12eV/Å.

A Correlation between samples produced by molecular dynamics

$\mathbf{X}$ is generated using molecular dynamics. In each step of molecular dynamics, the force exerting on each atom is calculated using the EMT potential, which, according to Newton's law, updates its velocity, which in turn affects the trajectory of the atom. As the atoms move in continuous trajectories, $\mathbf{X}_\alpha$ and $\mathbf{X}_{\alpha+1}$ are naturally correlated. The one-sample-per-50-timestep rule alleviates the problem so that $\mathbf{X}_\alpha$ is no longer almost the same as $\mathbf{X}_{\alpha+1}$, but the correlation is far from being completely eliminated. Figure 5 shows $Cov(\mathbf{X}_\alpha, \mathbf{X}_{\alpha+\delta})$ at different δ's.

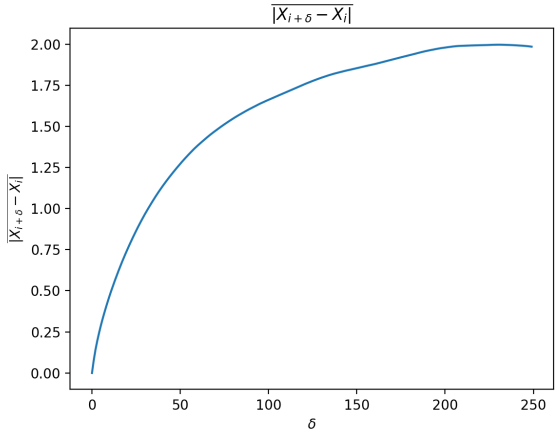


Figure 5: $Cov(\mathbf{X}_\alpha, \mathbf{X}_{\alpha+\delta})$

This incurs problems if training and test sets are chosen randomly. If sample 1 and 3 are in training set and sample 2 is in test set, label 2 can be easily obtained by remembering and averaging labels 1 and 3. This increases the test accuracy spuriously.

B Preprocessing data for the manybody expansion model

Let's denote the entire dataset as $\underset{\text{Nsample}\times\text{Natom}\times 3}{\mathbf{X}} \to \underset{\text{Nsample}}{\mathbf{E}}$. Let α index samples, and i, j, k index atoms (*e.g.* manybody expansion is $\sum_i, \sum_{ij}, \sum_{ijk}$).

Consider the choice of features for describing 2-body interactions. A function that describes 2-body interaction can be simplified using the fact that E is translationally and permutationally invariant, reducing the number of features from 6 to 1:

$$f(\overrightarrow{x_i}, \overrightarrow{x_j}) \to f(r_{ij})$$

and so can the 3-body interaction function (9 to 3 features):

$$f(\overrightarrow{x_i}, \overrightarrow{x_j}, \overrightarrow{x_k}) \to f(r_{ij}, r_{jk}, r_{ik})$$

The 2-body interactions feature tensors $\underset{\text{Nsample}\times\text{Natom}\times\text{Natom}}{\mathbf{R}}$, $\underset{\text{Nsample}\times\text{Natom}\times\text{Natom}}{\mathbf{C}^2}$ are defined as:

$$\mathbf{R}_{\alpha ij} = r_{ij},\ \mathbf{C}^2 = 1/\mathbf{R} \text{ (element-wise)}$$

We are inverting $\mathbf{R}$ because, due to the local nature of E, greater $\mathbf{R}$ means less significant interactions, and we want to emphasize smaller $\mathbf{R}$.

When inverting $\mathbf{R}$, where $\mathbf{R}_{\alpha ij} = 0$, we set $\mathbf{C}^2_{\alpha ij} = 0$. The reason is that $\mathbf{R}_{\alpha ij} = 0$ means $i = j$, and thus the term describes the 2-body interaction between the atom and itself; it doesn't exist (1-body interaction is described by other feature tensors). Thus $\mathbf{R}_{\alpha ij} = 0$ is the same case as where two atoms are infinitely far apart, $\mathbf{R}_{\alpha ij} = \infty$, and thus it's appropriate to set $\mathbf{C}_{\alpha ij}$=0.

We then standardize $\mathbf{C}^2$. Note that $\mathbf{C}^2_{\alpha ij} = 0$ has a special meaning and standardization destroys it. However, we also tested normalization to (0,1), and as expected, it leads to very poor training of neural networks.

Finally, 3-body interactions' feature tensors $\underset{\text{Nsample}\times\text{Natom}\times\text{Natom}\times\text{Natom}\times 3}{\mathbf{C}^3}$

$$\mathbf{C}^3_{\alpha ijk1} = \mathbf{C}^2_{\alpha ij}, \mathbf{C}^3_{\alpha ijk2} = \mathbf{C}^2_{\alpha ik}, \mathbf{C}^3_{\alpha ijk3} = \mathbf{C}^2_{\alpha jk}$$

$\mathbf{E}$ and $\mathbf{F}$ are also standardized.

References

[1] James B Hendrickson. Molecular geometry. i. machine computation of the common rings. *Journal of the American Chemical Society*, 83(22):4537–4547, 1961.

[2] WMC Foulkes, L Mitas, RJ Needs, and G Rajagopal. Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33, 2001.

[3] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

[4] John P Perdew. Climbing the ladder of density functional approximations. *MRS bulletin*, 38(9):743–750, 2013.

[5] KW Jacobsen, JK Norskov, and Martti J Puska. Interatomic interactions in the effective-medium theory. *Physical Review B*, 35(14):7423, 1987.

[6] Thomas B Blank, Steven D Brown, August W Calhoun, and Douglas J Doren. Neural network models of potential energy surfaces. *The Journal of chemical physics*, 103(10):4129–4137, 1995.

[7] Jörg Behler. First principles neural network potentials for reactive simulations of large molecular and condensed systems. *Angewandte Chemie International Edition*, 56(42):12828–12840, 2017.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[11] Jörg Behler. Neural network potential-energy surfaces for atomistic simulations. *Chemical Modelling: Applications and Theory*, 7(1), 2010.

[12] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, et al. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

[13] Siva Chiriki, Shweta Jindal, and Satya S Bulusu. Neural network potentials for dynamics and thermodynamics of gold nanoparticles. *The Journal of chemical physics*, 146(8):084314, 2017.

[14] Wendy D Cornell, Piotr Cieplak, Christopher I Bayly, Ian R Gould, Kenneth M Merz, David M Ferguson, David C Spellmeyer, Thomas Fox, James W Caldwell, and Peter A Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117(19):5179–5197, 1995.

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

[16] François Chollet et al. Keras. `https://keras.io`, 2015.

[17] Siva Chiriki, Shweta Jindal, and Satya S Bulusu. Neural network potentials for dynamics and thermodynamics of gold nanoparticles. *The Journal of chemical physics*, 146(8):084314, 2017.