Feasibility Study of Graph Neural Networks with Atomic Cluster Expansion

Minsik Cho

May 15, 2023

1 Introduction

The field of computational chemistry has a fundamental cost-accuracy tradeoff that limits accurate simulation of large systems in a long time scale. Because of the complexity of solving the Schrödinger Equation even in the time-independent regime (and even more so in the time-dependent case), various approximation techniques have emerged from the birth of the field. As shown in Figure 1[4], simulation of longer time scales or larger system size necessitate more aggressive approximations, which may often leave out important details only captured in more precise levels of calculation.

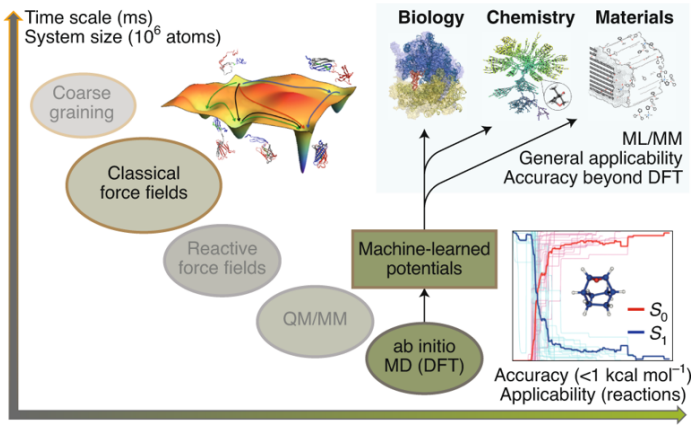

Figure 1: Tradeoff between accuracy and time scale (or system size)

One interesting point to note here is that *ab initio* Molecular Dynamics that use Density Functional Theory (DFT) backend is the most *accurate* method of choice in the plot. This is in fact the only method in the plot that fully depicts the system based on quantum mechanical principles. For quantum chemical simulations that do not involve time evolution, this plot continues towards the lower right – in fact, DFT is regarded as less accurate but scalable choice in the

time-independent regime. It suffices to say that the prohibitive computational cost of more accurate methods limit the scientific exploration of longer-time phenomena, and without a major breakthrough in method development, such limitation will continue to be the biggest inhibitor. Machine-learned potentials, as shown in Figure 1, present an alternative to the tradeoff by estimating the high-quality quantum mechanical energies and forces using a pre-trained model. For interpolative estimations, which is often the case with the time evolved updates in molecular dynamics (MD) simulations, machine-learned potentials allow fast and yet accurate scalable approach. Machine-learned Potentials (MLPs) thus provides a pathway towards larger problems in *ab initio* molecular dynamics (AIMD) that were previously deemed as intractable due to the prohibitive computational complexity.

MLPs have already seen numerous applications in biochemistry and material science[2]. For force field applications, the benefits are especially huge as it allows treatment of solvent material[9] at a relatively low cost. Compared to dielectric continuum used for most quantum mechanical solute calculations, MLPs provide some quantum mechanical details of the bath solvent at a reasonable computational cost.

At the Center for the Exascale Simulation of Materials in Extreme Environments (CESMIX), there are on-going efforts in development of machine-learned interatomic potentials (MLIAPs) using high-level programming language, such as `Julia`. Utilizing the software composability of `Julia` and differentiable programming provided by `Zygote`, different strategies are explored at CESMIX to further advance the state-of-the-art MLIAPs.

In this final project, a feasibility study of Graph Neural Networks (GNNs) with Atomic Cluster Expansion (ACE) for MLIAPs was conducted. Noting from the fact that molecules (and its geometries) are inherently graphs, GNNs have been viewed as a natural choice of neural network architecture for molecular applications[7]. Convolutions in GNNs, such as localized convolutions, also match the chemists' intuition that interaction between near pairs contribute more. Long range effects are much smaller and are relatively insignificant on the level of DFT (and DFT-based MLIAPs). Because ACE is one of the most popular approaches in MLIAP development, a natural motivation is to attempt use GNNs with ACE.

2 Graph Neural Networks

In graph neural networks (GNNs), a key quantity to start the discussion with is the adjacency matrix. Suppose that we have a graph shown in Figure 2. For this graph, we can define an adjacency matrix $A \in \mathbf{N}^{n\times n}$ where n is the number of nodes in the graph. Each element of the adjacency matrix A_{ij} is then the number of edges that connect the i-th and the j-th nodes. For the graph above,

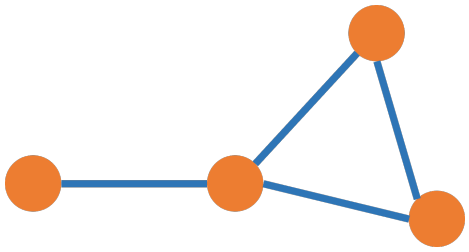

Figure 2: Example of a graph

the adjacency matrix is

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Note that the adjacency matrix by construct has some nice qualities as a representation for molecules. First, permutation of the order of constituent atoms simply correspond to permutation of basis in the adjacency matrix, which does not alter the information represented in the matrix. Second, it allows a rotationally invariant representation – the molecular input does not change with trivial overall rotation. Adjacency matrix does have a few issues, namely normalization and distance thresholding. The adjacency matrix, in practice, is normalized to prevent from the training process to explode, as recommended in a paper[6]. Arbitrary distance threshold applied to determine whether two atoms are *connected* or not cannot be avoided trivially, but in many cases (where there is a clear cut distinction between the two) this is not a prohibitive concern. Some ideas to circumvent the arbitrary thresholds will be discussed in the later sections as a future directions.

Now that the input graphs are defined, neural network models are formed with attributes associated with the components of the graph. Each node, edges, and global (entire graph) context can have corresponding attributes. For the case of molecules, information about the atoms (which will be the local descriptors in Atomic Cluster Expansion) are the node attributes, and the estimated (or the true) energies are the global attributes. With these information, GNNs are usually defined with some combination of graph convolutions and pooling functions[1], such as sum operation over all perceptrons. By optimizing the parameters that feed into the neural network, models are trained using input dataset.

3 Atomic Cluster Expansion

Atomic Cluster Expansion (ACE) is a cluster expansion technique that expresses energies and forces as a many-body expansion of atoms within some cutoff

distance[3]. With ACE, energy contribution of atom i is expressed as

$$E_i = \varepsilon^{(1)}_{(i)} + \frac{1}{2}\sum_j \varepsilon^{(2)}_{(i,j)} + \frac{1}{6}\sum_{j,k} \varepsilon^{(3)}_{(i,j,k)} + \cdots \tag{1}$$

Here, the indices $j, k, \cdots$ run over the neighboring atoms within some threshold distance.

In obtaining each cluster term, ACE formalism starts by defining some neighborhood density using Dirac-delta functions with peaks on neighboring atoms within the threshold distance.

$$\rho(i,\mu) = \sum_{j\neq i} \delta_{\mu,\mu_j}\delta(r - r_{ji}) \tag{2}$$

Here, μ denotes the elemental identity (i.e. number of protons) of the corresponding atom. Then, the neighborhood density is projected into hydrogen-like harmonic orbitals.

$$\phi_{\mu_i,\mu_j,n,l,m} = R^{\mu_i,\mu_j}_{n,l}(r_{ji})Y_{l,m}(r_{ji}) \tag{3}$$

$$A_{i,(\mu_i,\mu_j,n,l,m)} = \langle \rho(i,\mu) | \phi_{\mu_i,\mu_j,n,l,m}\rangle \tag{4}$$

$$= \sum_j \phi_{\mu_i,\mu_j,n,l,m}(r_{ji}) \tag{5}$$

Finally, tensor products for n-th order cluster terms are taken and are used as basis functions.

$$A_{i,\nu} = \prod_{t=1}^{\nu} A_{i,(\mu_i,\mu_j,n,l,m)} \tag{6}$$

$$\psi^{(p)}_i = \sum_\nu c^{(p)}_\nu B_{i,\nu} \tag{7}$$

$$= \sum_\nu c^{(p)}_\nu \sum_{\nu'} c_{\nu,\nu'} A_{i,\nu} \tag{8}$$

$$E_i = \mathcal{F}(\psi^{(0)}_i, \cdots, \psi^{(p)}_i, \cdots) \tag{9}$$

This representation, which is the key idea of Atomic Cluster Expansion (ACE) retains rotational, translational, and permutation invariance. Its use of hydrogenic harmonics allows a path toward easier analyses by chemists. For GNNs, these expansion terms that are calculated for each constituent atom are the ACE local descriptors that comprise the node attributes in the neural network.

4 Method

Benefiting from the software composability of `Julia` and the existing code base maintained by CESMIX, the general workflow of the initial test of GNN based on ACE formalism operates as listed below:

1. Read in training dataset (aHfO2, SNAP) using the `ExtXYZ` struct defined in `PotentialLearning.jl` – Molecular geometries and DFT energies are loaded.
2. Evaluate the ACE local descriptors using `ace.jl`
3. Build normalized adjacency matrix for periodic cells
4. Define a GNN model based on convolutions of graphs using `GraphNeuralNetworks.jl`
5. Train neural net using `Flux.jl`, which performs automatic differentiation (AD) to obtain the Jacobian of the parameters used in GNN model
6. Compare the loss and time performance with linear (least-squares) solution

For the adjacency matrix build step, distance threshold of 0.5 Å was applied after taking the periodicity into account (by using the crystal lattice vectors) and the matrix was normalized as:

$$A' = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{10}$$

where D is the degree matrix, a diagonal matrix with the total number of edges from the i-th node. Use of an one-hot vector for the elemental identity was tested by creating `struct OneHotAtom` that extends the `AbstractVector`. However, its effect on the learning process, compared to just relying on the ACE local descriptor (which embeds the elemental identity) was minimal and was therefore removed moving forward.

a-HfO2 dataset[8], which consists of hafnium dioxide systems with 96 atoms (which are either hafnium or oxygen) in periodic cell, was used as test dataset.

5 Results

5.1 Initial Trial

For the initial trial of the GNN/ACE implementation for machine-learned energies, the learning rate was set to a very conservative ($= 10^{-4}$) value to assess the convergence pattern. Without proper treatment of the adjacency matrix (through normalization and periodic treatment), the training of the neural network exploded after the first few epochs, so this *initial* trial refers to the first run after the critical issues were fixed.

In the linear least-squares fit of 200 train data and 100 test data, mean squared error (MSE) was 0.10 eV. Linear fit took about 67 seconds.

As shown in Figure 3, initial test of GNN/ACE showed a promising result where the loss function gradually decreased in each epoch. After about 35000 epochs, it reached about 0.37 eV MSE, resulting in a value comparable to the least squares approach.

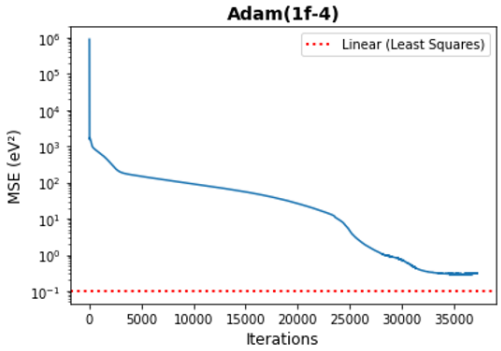

Figure 3: Initial Trial of GNN/ACE Implementation

Performance-wise, the whole training required a long time, in the orders of hours on a 20-core CPU. GPU programming was not employed at this point, because the number of training dataset was small enough.

For a faster convergence, for subsequent calculations, a larger learning rate was used. Most commonly used scheme (will be specified in detail below) was to take a learning rate of $\eta = 0.1$ and decay of momentums $\beta = (0.9, 0.8)$ for the first few (about 500) epochs and a slow rate ($\eta = 1e^{-5}, \beta = (0.9, 0.8)$) for the later iterations. Calculations in the next subsections took in the order of minutes, lowering the computational cost of training the neural net comparable to solving the least squares problem.

5.2 Dependence on the Number of Training Data

To examine how the training of GNN/ACE neutral net behaves with a different number of training dataset, training was performed for 25, 50, 100, and 200 dataset.

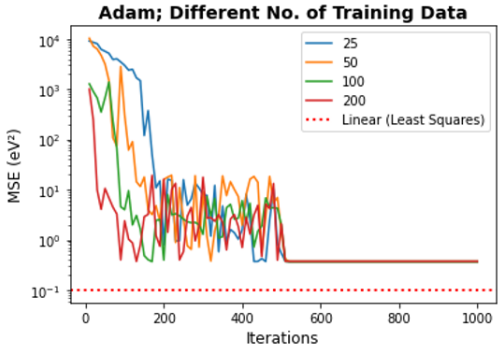

Figure 4: Training with Different Numbers of Training Data

As shown in Figure 4, using different number of training data did not have a large impact on the converged MSE. It is encouraging to observe that convergence to about 0.30 eV was reached in a similar number of epochs. Larger number of training data of course lengthens each epoch, of course, but this hints a nice scalability of the GNN/ACE model.

5.3 Activation Function Choices

Different choices of activation functions were also tested to identify the characteristics of the GNN/ACE model.

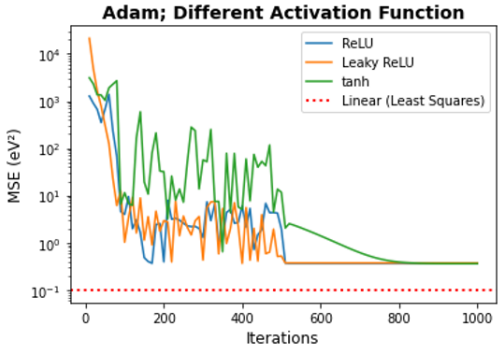

Figure 5: Training with Different Activation Functions

As shown in Figure 5, use of different activation functions in the first convolution layer did not affect the finally converged value by much. Hyperbolic tangent (fast tanh in `Flux.jl`) function has a longer tail compared to ReLU and Leaky ReLU, which share similar convergence behavior, but the differences are relatively insignificant.

6 GPU Implementation

GPU version of the implementation was trivially written using the `Flux.gpu` binding to CUDA libraries. For simple neural networks only consisting of two convolutional layers and one pooling & dense layer, the V100 GPU did not provide significant speedup compared to the CPU setup (20 CPU cores reserved on Intel Xeon-based computing cluster). It is expected that production level training would require computations on a GPU node.

7 Conclusions

Through this final project, the immediate goal of testing the applicability of graph neural networks (GNNs) and atomic cluster expansion (ACE) has been

achieved. GNN/ACE provides comparable accuracy to linear squares solution, and mean squared error of the GNN/ACE model is well within the target values MLIAPs aim for[1]. Larger and more complicated neural net models are being tested using GPU implementation as of writing this report. It is expected that GNN/ACE may provide a MLIAP that preserves the chemists' intuition embodied in the GNN and ACE formalism.

The graph-based formalism of GNN also provides some opportunities to bridge in graph-based methods in quantum chemistry. Bootstrap Embedding[10] (BE), which is a kind of QM/QM [2], evaluates fragment-by-fragment energies to recover the energy of the whole chemical system. These fragments, which are two (in BE2), three (in BE3), or n (in BEn) atoms next to each other in a chemical systems, are inherently found in adjacency matrices used in GNNs. Therefore, BE energies can provide edge level (from BE2) features that MLIAPs can be trained against instead of graph level features (total energies). Previous attempts in using the edge features for molecular application only use edge features as descriptors[5], but using BE energies allow us to fully exploit edge level structure to actual estimator outputs. Because of the radically lower computational cost of BE calculations, reference data can also originate from much higher-level quantum mechanical calculations. Instead of mean-field level calculations (i.e. Density Functional Theory), correlated calculations (i.e. Coupled Cluster) can be performed routinely. Generalizations to BEn ($n \geq 3$) is less clear, since each fragment represents more than a single edge.

In conclusion, this feasilbility study of GNN/ACE have shown the possibility of a new approach in MLIAP development. Interesting opportunities stemming from the nice data structure of graphs also present new directions in MLIAP development.

[1] Unlike quantum chemistry methods, which target chemical accuracy – 1kcal/mol – MLIAPs take per atom measures.

[2] quantum mechanical method embedded in another quantum mechanical method

References

[1] Nikolas Adaloglou. How graph neural networks (gnns) work.

[2] Jörg Behler and Gábor Csányi. Machine learning potentials for extended systems: a perspective. *The European Physical Journal B*, 94(142), 2021.

[3] Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Phys. Rev. B*, 99:014104, Jan 2019.

[4] Pascal Friederich, Florian Häse, Jonny Proppe, and Alán Aspuru-Guzik. Machine-learned potentials for next-generation matter simulations. *Nature Materials*, 20(6):750–761, 2021.

[5] Liyu Gong and Qiang Cheng. Exploiting edge features in graph neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[7] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 2021.

[8] Ganesh Sivaraman, Anand Narayanan Krishnamoorthy, Matthias Baur, Christian Holm, Marius Stan, Gábor Csányi, Chris Benmore, and Álvaro Vázquez-Mayagoitia. a-hfo2 dataset: "machine-learned interatomic potentials by active learning: amorphous and liquid hafnium dioxide". *npj Computational Materials*, 2020.

[9] Adri C. T. van Duin, Siddharth Dasgupta, Francois Lorant, and William A. Goddard. Reaxff: A reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, 105(41):9396–9409, 2001.

[10] Hong-Zhou Ye and Troy Van Voorhis. Atom-based bootstrap embedding for molecules. *Journal of Physical Chemistry Letters*, 10:6368–6374, 2019.

Appendix

Source code for the GNN/ACE implementation can be found in this pull request on `PotentialLearning.jl` Github repository. This final project was conducted with Dr. Emmanuel Lujan's (Center for the Exascale Simulation of Materials in Extreme Environments) guidance.