Embedded Atom Neural Network Potential Fitting Package User Guide

-B. Jiang Group (2019)

Method

The embedded atom neural network (EANN) approach is inspired by the well-known empirical embedded atom method (EAM) model[1-2]. The EAM approach originates from the quasi-atom[3] (or equivalently effective medium[4]) theory in which the embedding energy of an impurity in a host is a functional of the electron density of the unperturbed host without impurity, *i.e.* $E_i = F\left[\rho(\mathbf{r})\right]$. Assuming that the impurity experiences a locally uniform electron density[3], the embedding energy can be approximated as a function of the scalar local electron density at the impurity site plus an electrostatic interaction[1-2, 5].

Considering all atoms in the system as impurities embedded in the electron gas created by other atoms, in the EAM framework, the total energy of an N-atom system is the sum over all individual impurity energies[1-2], that is:

$$E = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N}\left[F_i(\rho_i) + \frac{1}{2}\sum_{j\neq i}\phi_{ij}(r_{ij}) \right] \quad (1)$$

Where F_i is the embedding function, ρ is the embedded electron density at the position of atom i given by the superposition of the densities of surrounding atoms, ϕ_{ij} is the short-range repulsive potential between atoms i and j depending on their distance r_{ij}. The EAM or even its modified version[6] has limited accuracy and is mainly suitable for

metallic system.

To beyond the EAM, we need to improve both expression of the embedded density and the function F. In this EANN approach, we start from the commonly used Gaussian-type orbitals[7] (GTOs) centered at each atom,

$$\varphi_{l_x l_y l_z}^{\alpha, r_s} = x^{l_x} y^{l_y} z^{l_z} \exp\left(-\alpha \left| r - r_s \right|^2\right) \tag{2}$$

$\mathbf{r}=(x, y, z)$ constitutes the coordinate vector of an electron, r is the norm of a vector, α and r_s determine radial distributions of atomic orbitals, $l_x+l_y+l_z=L$ specifies the orbital angular momentum (L), *e.g.* L=0, 1, 2, correspond to the *s*, *p* and *d* orbitals, respectively. Thus, the embedded density of atom i can be taken as the square of the linear combination of atomic orbitals from neighboring atoms. This would generate a scalar ρ^i value for the embedding atom i, as used in the EAM approach, which has been proven to offer insufficient representability for the total energy and can be improved by including the gradients of density[3]. Here, we alternatively evaluate individual electron density contribution from the same type of GTOs with the equivalent L, α and r_s, namely,

$$\rho_{L,\alpha,r_s}^{i} = \sum_{l_x,l_y,l_z}^{l_x+l_y+l_z=L} \frac{L!}{l_x!\, l_y!\, l_z!} \left(\sum_{j=1}^{n_{atom}} c_j \varphi_{l_x,l_y,l_z}^{\alpha,r_s}\left(\mathbf{r}_{ij}\right) \right)^2 \tag{3}$$

where r_{ij} represents the Cartesian coordinates of the embedded atom i relative to atom j (or the coordinates of the electron at the position of atom i), c_j is the expansion coefficient of an orbital of atom j, or equivalently an element-dependent weight[8] that is optimized in the training process, the factorials of l_x, l_y, l_z and L constitute a pre-factor

that enables the convenient transformation from Eq.(3) to an angular basis as realized by Takahashi[9], n_{atom} counts the number of neighboring atoms close to the embedding atom within a sphere with a cutoff radius (r_c)[10], A cosine type cutoff function[10] is also multiplied to each orbital here to decay the interaction to zero smoothly approaching r_c.

Next, the embedding function F can be replaced with an atomistic neural network (AtNN). Since the core-core repulsion $\phi_{ij}(r_{ij})$ depends also on the internuclear distance r_{ij}, this part of contribution has been automatically incorporated in the NN and require no extra terms, Eq. (1) can be thus rewritten as，

$$E = \sum_{i=1}^{N} E_i = \sum_{i=1}^{N} NN_i\left(\boldsymbol{\rho}^{\boldsymbol{i}}\right) \tag{4}$$

Where $\boldsymbol{\rho}^i$ consists of the orbital-dependent density components defined in Eq. (3). This naturally gives us the equivalent AtNN representation[11] in which $\boldsymbol{\rho}^i$ represents a set of local atomic density descriptors. As a result, the EANN method overcomes the intrinsic approximations on the uniform density and pairwise superposition of scalar density contributions in the EAM. It hence enables the accurate reproduction of the ab initio total energy and is no longer limited to metallic system.

Reference

1. Daw, M. S.; Baskes, M. I. Semiempirical, Quantum Mechanical Calculation of Hydrogen Embrittlement in Metals. *Phys. Rev. Lett.* **1983,** *50* (17), 1285-1288.

2. Daw, M. S.; Baskes, M. I. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Phys. Rev. B* **1984,** *29* (12), 6443-6453.

3. Stott, M. J.; Zaremba, E. Quasiatoms: An approach to atoms in nonuniform electronic systems. *Phys. Rev. B* **1980,** *22* (4), 1564-1583.

4. Nørskov, J. K.; Lang, N. D. Effective-medium theory of chemical binding: Application to chemisorption. *Phys. Rev. B* **1980,** *21* (6), 2131-2136.
5. Zhang, Y.; Hu, C.; Jiang, B. Embedded Atom Neural Network Potentials: Efficient and Accurate Machine Learning with a Physically Inspired Representation. *J. Phys. Chem. Lett.* **2019,** *10* (17), 4962-4967.
6. Baskes, M. I. Modified embedded-atom potentials for cubic materials and impurities. *Physical review. B, Condensed matter* **1992,** *46* (5), 2727-2742.
7. Boys, S. F. Electronic wave functions - I. A general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London Series a-Mathematical and Physical Sciences* **1950,** *200* (1063), 542-554.
8. Gastegger, M.; Schwiedrzik, L.; Bittermann, M.; Berzsenyi, F.; Marquetand, P. wACSF—Weighted atom-centered symmetry functions as descriptors in machine learning potentials. *J. Chem. Phys.* **2018,** *148*, 241709.
9. Takahashi, A.; Seko, A.; Tanaka, I. Conceptual and practical bases for the high accuracy of machine learning interatomic potentials: Application to elemental titanium. *Phys. Rev. Mater.* **2017,** *1* (6).
10. Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **2011,** *134* (7), 074106.
11. Behler, J.; Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **2007,** *98* (14), 146401-4.
12. Zhang, Y.; Hu, C.; Jiang, B. Embedded Atom Neural Network Potentials: Efficient and Accurate Machine Learning with a Physically Inspired Representation. J. Phys. Chem. Lett. 2019, 10 (17), 4962-4967.

How to Use EANN Package

EANN package was written by Fortran and paralleled by Openmp standard with shared-memory architecture. Most of its matrix-vector and matrix-matrix operations are completed by high-performance Math Kernel Library (MKL). Users can employ geometries, energies and atomic force vectors (or some other physical properties which are invariant under rigid translation, rotation and permutation of identical atoms and its corresponding gradients) to construct a model to represent these physical properties via this package. There are three routines to use this package:

1. Prepare data

2. Construct a model

3. Employ the model

1. Prepare data

There are three files that users need to prepare, namely, “input”, “1” and “atom_1” files. Note that “1” and “atom_1” are put in the folder named “data”. “input” contains all parameters that fitting required, “1” contains the coordinates, potential energies and atomic force vectors of all calculated points for the first system. “atom_1” contains the names of all atoms that is consistent with the order of atoms in file “1”. If there are systems more than one in one fitting, additional files should be prepared, e.g. “2”, “atom_2”, “3”, “atom_3”, etc..

For example, users want to represent three systems (CH_4, CH_3CH_3, $CH_3CH_2CH_3$)

in one unified model. Users need to prepare files ‘1’, ‘2’, ‘3’ to hold coordinates of CH_4, CH_3CH_3 and $CH_3CH_2CH_3$ respectively and files ‘atom_1’, ‘atom_2’ and ‘atom_3’ for the elements of corresponding systems with the following format.

The format of “1” / “2”/ “3” file in “data” folder

100.0 0.0 0.0

0.0 100.0 0.0

0.0 0.0 100.0

point= 1 5.06245

0.46543419 0.81826725 0.59965180 -4.614025 -1.101006 0.539959

0.39050893 0.76541778 0.62423227 -0.305477 -0.434224 0.073952

0.60106142 0.81463284 0.62123169 -2.503968 -0.309251 0.421334

0.51270681 0.78621015 0.55395716 -1.727109 -1.045655 1.575540

0.42355461 0.89759874 0.60379835 -0.080189 -0.187667 0.566722

point= 2 5.05245

0.60553419 0.31826725 0.90465180 -6.614025 1.101006 -0.539959

0.39050893 0.76541778 0.62423227 -0.305477 0.434224 -0.073952

0.60106142 0.81463284 0.62123169 -2.503968 0.309251 -0.421334

0.51270681 0.78621015 0.55395716 -1.727109 1.045655 -1.575540

0.42355461 0.89759874 0.60379835 -0.080189 0.187667 -0.566722

…

Top three lines are the lattice vectors defining the unit cell of the system. Users can copy the lattice vectors in POSCAR to this file. If the studied system is not periodic, users should employ a large enough to eliminate the period. Next line: Arbitrary characters, Arbitrary number, Potential energy Following lines: First three columns are the coordinates of the geometry, the last three columns are atomic force vectors (if you don't want to incorporate the force vector in you training, these three columns can be omitted). The atoms allowed to move should be placed in Front lines.

The format of "atom_1" / "atom_2" / "atom_3" file in "data" folder

C T

H T

H T

This file has two columns. First one is the atom names and the order must be consistent with the "1" file. Second one is a character "T" or "F", "T" means the atom was allowed to move in the ab initio calculations, "F", whereas, conversely.

2. Construct a model

There is some brief description about the EANN in the method section and more details can be found in ref 12. In former section "Prepare data", file '1' and 'atom_1' have informed the package the information about the system. However, some hyper-

parameters about the embedded density, training algorithms and architectures of neural networks are essential for obtaining an exact representation. Next, we will give an example to explain these hyper-parameters in detail.

Parameters in "input" file

1. 1 # start_force

(If the atomic force vectors are used in NN training. "1": energies and force vectors will be used; "0": only energies will be used.)

2. 1 # start_wb

(Training algorithm, "1": hybrid extreme machine learning and Levenberg-Marquardt (ELM-LM) algorithm(best); "0": LM algorithm.)

3. 0 # start_init

(Initialize parameters for a model, "0": general method; "1": restart from a previous training)

4. 1 # table_coor

(The form of the coordinates in "1" file, "0": Cartesian coordinates. "1": Fraction coordinates.)

5. 1 # table_grid

("0": handwrite each r_s; "1": generate even grids r_s by the given β.)

1th ~ 5th are all switch parameters. "**Start_force**" is used to control if the atomic force vectors are employed to construct models or not. "1" is strongly recommended to utilize the gradient information for that it can reduce the required numbers of configurations

dramatically. "**Start_wb**" is a parameter to choose the training algorithm. ELM-LM algorithm is developed based on the LM algorithm, which can accelerate the training process without any harm to the accuracy. "**table_grid**": If users are not familiar with density-like descriptors, "1" with β = 0.1~0.3 is a proper choice. "0" is an exploitation option for experts, which can write each density center r_s and corresponding gaussian broaden α. Actually, the automatically generated even grids can obtain a satisfied result for most system according to our tests.

6. 1d-2 # baise_rmse

"baise_rmse": convergence criteria, if decreases of cost function of continuous six steps are all less than the criteria, we believe this fitting is convergent and this training will stop.

7. 2 # maxnumtype

(Number of different element types.)

8. 5 5 # maxnumatom maxneff

(Total number of atoms and number of atoms allowed to move)

9. 1 # nsurf

(Number of system that user wants to fit at the same time.)

Note: "maxnumtype" is total number of chemical elements in all systems. Not the maximal chemical elements of all systems. However, "maxnumatom" and "maxneff" are the maximal atoms and effective atoms of all systems. "nsurf" is a parameter that represent the number of systems you want to include in one unified model.

10. 2 # ipsin

(The orbital angular momentum (L) in Eq.(3))

11. 9 # maxnwave

(Maximal number of the GTOs and should equal the number of r_s. Note: actual number is maxnwave+1)

12. 9 9 # nwave

(Number of the GTOs for each element and should equal the number of r_s. Note: actual number is nwave+1)

These parameters are employed to generated the embedded density vectors, which are key parameters for constructing an exact representation. **"ipsin"** is L that determines the order of angle functions. "ipsin = 2" means L = 0, 1, 2 and corresponds to the s, p and d orbitals. "**nwave**" is number of the GTOs-like radial functions for each element. **"maxnwave"** is the maximum numbers of GTO-like radial functions of all chemical elements. As far as we known, the evaluation of embedded density vectors is the time-consuming step and "nwave" and "ipsin" are the essential parameters that determined the quality and quantity of evolution of embedded density, which need to be tested to find a balance between the accuracy and efficiency.

13. 100, 100 # cutnum, scutnum

Cutnum: maximal number of atoms in the cutoff distance; Scutnum: maximum number of atoms in each cube with side length rc

14. 5 # ncycle

(Number of fitting in a submitted job)

15. 10000 # nloop

(Maximum iterations in one fitting.)

16. 1 # nbatch

"nbatch" controls the required memory during fitting. Jacobian matrix will be divded into nbatch sub-Jacobain matrixes for saving memory. However, a too large "nbatch" will result in the decrease of parallel efficiency at the same time.

17. 1000 1000 # numpoint maxnpoint

(numpoint: total number of the geometries used in the fitting. maxnpoint: maximum number of geometries of all systems.)

18. 900 # maxtpoint

(Number of geometries in training set, in general, 90% of the total number of points are selected.)

19. 1d0 1d0 # perindex force_perindex

(Weight of energy (W_E) and force (W_F) in fitting)

The realistic W_F equals $\sqrt{force\_perindex / 3.0 \times \text{neff}}$.

20. 6d0 6d0 # rc

(Cutoff distance for each element.)

21. 60 # mnl

(Maximum number of neurons for input and hidden layers.)

22. 2 # mnhid

(Maximum number of the hidden layers)

23. 1 # nkpoint

(Number of neural networks with same structures in the fitting.)

24. 1 # outputneuron

(Number of neurons in the output layer.)

25. 'C' 2 30 60 0.2 # atomtype

(Element name, number of hidden layers for the element, Number of neurons of the first hidden layer, Number of neurons of the second hidden layer. “β” mentioned in the introduction.)

26. 0 0.8 0.0 # alpha r_s

27. 1 0.8 0.5

28. 2 0.8 1.0

29. 3 0.8 1.5

30. 4 0.8 2.0

31. 5 0.8 2.5

32. 6 0.8 3.0

33. 7 0.8 3.5

34. 8 0.8 4.0

35. 9 0.8 4.5

if “Table_grid=0”: Label of lines, α and r_s in Eq. (3) until $r_s + \Delta r_s \geq r_c$, If Table_grid=1, 26-35 lines do not need be written, the program will produce the grid automatically

($\alpha_s = \beta / \Delta r_s^2$). Line 26-35 should be repeated for every element in the system. Users can use different neural network structure to train different elements.

With all needed file prepared, you need to compile these Fortran files to get an executable file. Note: MKL must be included in the compile. We have provided a 'makefile' file as an example. These Fortran files are linked in the "makefile", "make" to get the executable file "nn.exe" and copy this file to the work directory that contains "input" file and "data" folder.

Note: when you submit your job, you must add the 'export OMP_STACKSIZE=M' (M ~ 100000 and increase with the number of NN parameters) in you script for allocating enough memories for openmp threads.

In the training process, some iterative information will output to screen and you can assign it to a file, for example "./nn.exe > nn.out".

- nn.out

First, some information about the model will be printed out.

```
Fitting the PES based on the atomic neural network
Based on the idea of embeeding atomic method
the total parameter: allnw=       11610
the total wave parameter: wave_allnw=         540
number of configurations used for training       12909
number of total target:     1600716
number of AtNN           1
number of neuron of output layer           1
 1.00000000  1.00000000  0.00010000
     9    65   123 14343
C    60   30   60    1
H    60   30   60    1
Ir   60   30   60    1
```

After that, RMSE of energy and values of cost function of every iterations will be

printed out.

Energy RMSE for training data Energy RMSE for validation data

```
2019/09/08  06:11:22.37
Epoch      1/ 10000; Mu= 1.E-03; Perform= 5.046E+01 2.211E+02 / 5.128E+01 2.296E+02 meV; Val= 0/ 6
Epoch      2/ 10000; Mu= 1.E-03; Perform= 4.281E+01 2.091E+02 / 4.300E+01 2.117E+02 meV; Val= 0/ 6
Epoch      3/ 10000; Mu= 1.E-04; Perform= 4.278E+01 1.968E+02 / 4.331E+01 2.014E+02 meV; Val= 1/ 6
Epoch      4/ 10000; Mu= 1.E-04; Perform= 3.559E+01 1.734E+02 / 3.582E+01 1.692E+02 meV; Val= 0/ 6
Epoch      5/ 10000; Mu= 1.E-04; Perform= 3.238E+01 1.426E+02 / 3.280E+01 1.505E+02 meV; Val= 0/ 6
```

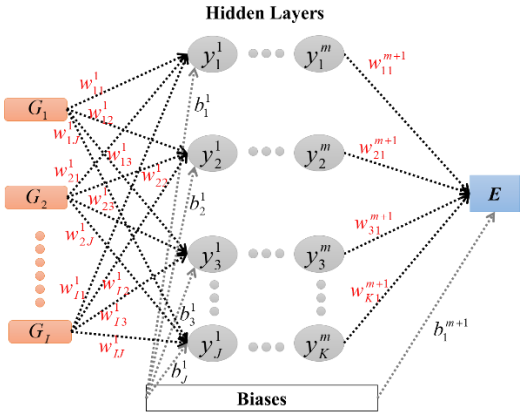
Values of cost funtions for training and validation

```
the rmse has converged
Total rmse_01=  18.7669 meV; Total force rmse_=  51.8094 meV/angstrom; train=  18.1950; validation=
 23.2911force train=  51.5189; force validation=  54.3543
```

These lines mean that a fitting have completed and parameters of model will output into the following files.

- W00*_[element name]1

Parameters of neural network for each element



The sequence of parameters are "w^1", "b^1", " ", "w^2", "b^2", " " , …, "w^{m+1}", "b^{m+1}", " ".

- weight_[element name]_00*

Weights of GTOs for each element.

Each descriptor and each element pair (the elements of the central atom and neighboring atom) have different GTO weights. For each central atom, the sequence of parameters is $\left\{c_j^i \; j=1,\cdots, \text{maxnumtype}\right\} \; i=1,\cdots, n_{den}$. ***n*****$_{den}$ is the number of elements in embedded density vector.**

- outputs.txt-*

Input potential energies, fitted potential energies, energy differences for each geometry.

- output_force.txt-*

Input force vectors, fitted force vectors, force differences for each dimension of geometry.

Employ the model

With a trained model, users can employ the model to obtain the energies and atomic force vectors of configurations. Furthermore, the models can be interfaced into the MD package for atomic simulation. In order to employ the model, you need to copy the trained model parameters to the work directory, including "input", "W00*_[element name]", "scalfactor_[element name]_00*" and "weight_[element name]_00*" file. Additional files "cell", "atom" is also needed. "cell" contains the lattice parameter of the system. "atom" contains the atom names with the order of the input geometry.

Note: change the name of "W00*_[element name]" to W_[element name]1,

“scalfactor_[element name]_00*” to “scalfactor_[element name]”, “weight_[element name]_00*” to “weight_[element name]”.

The Fortran interface:

At the beginning, users need to compile the Fortran files to obtain an executable file. Then users need to call the subroutine ‘init_pes’ to do some initializations.

Then you can pass the coordinates of configurations by calling the subroutine.

″eann_out(table, start_force, coor, y, force,hess)”

1. “table”: The form of the coordinates. “0”: Cartesian coordinates. “1”: Fraction coordinates.

2. “start_force”: “1”: energies and force vectors output with the variable ‘y’ and ‘force’ in the subroutine respectively; “0”: only energies obtained;

3. “coor”: the coordinates of the configurations provided for the calculating;

4. ‘y’: the array holding the energy of the corresponding configurations;

5. ‘force’: the array holding the atomic force of the corresponding configurations with its dimension numforce (numforce is the number of atomic forces).

6 ‘hess’: the array holding the second derivate of energy with respect to the coordinates with its dimension numforce*numforce.

Finally, you need to call “deallocate_all” to release the memory of program.