TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Personal Programming Project

Construction of artificial neural network potentials for atomistic material simulation

Sudarsan Manjunatha Pai
Matrkl no : 65109

Supervisors : Dr. Aruna Prakash,Dr. Geralf Hütter

Motivation

- The reliability of molecular dynamics (MD) depends crucially on the accuracy of the underlying potential-energy surface (PES)
- During the last decade,Density functional theory (DFT) has been extensively used to predict such material properties.
- Eventhough DFT is accurate,it comes at great computational cost,which limits the applicability when a structure contains few hundreds of atoms.
- Empirical interatomic potentials are also being used ,which rely on too many parameters in case of complex systems.
- This parameterization problem can be overcome by using ML techniques to interpolate the potential energy surface from reference calculations.
- Apart from that,I wanted to learn more about neural networks,deep learning and other ML techniques and gain practical experience in this domain.

Tasks and aims of the Project

Aim:

- To implement Behler-Parrinello(BP) machine learning potentials using feedforward artificial neural networks(ANN) for the representation of atomic energies.

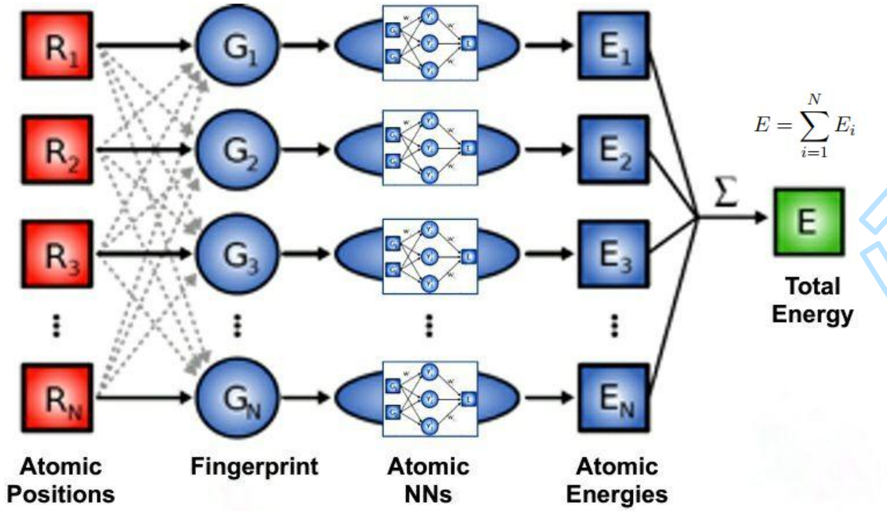
Proposed tasks:

- Conversion of raw data in the form of atomic co-ordinates to a fixed number of Behler-Parinello basis functions.
- Construction of atomic neural networks(with 70-N-N-1 architecture) from scratch with the help of OOP and numpy.
- Assembling the individual ANNs to form the high dimensional NNs and training the NN assembly in order to predict the structural energies.
- Testing and validation of all the functionalities with known inputs and outputs using pytest framework.

Additional tasks:

- Using the trained neural networks in Equation of states for the prediction of bulk modulus of rutile TiO_2.
- Using the neural network package in a molecular dynamics simulation program.

Details of the model implemented



Prediction of Bulk modulus

Molecular dynamics simulation

Force calculation → Application to MD

Pre-processing

- Data reading
- Data cleaning
- Converts local atomic environment into symmetry functions(structural fingerprint).

n atoms -->(n x 70) vector

Model development

- Implementing the High dimensional NN
- Hyperparameter tuning
- Simultaneous training of NNs in the NN assembly.
- Energy prediction

(n x 70) vector --> Energy

Force calculation

$$F_{k,\alpha} = -\frac{\partial E}{\partial R_{k,\alpha}}$$

Gradient of predicted energy found using the chain rule.

MD simulation

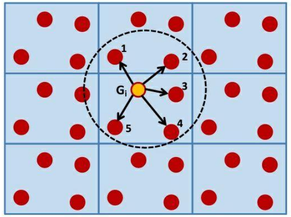
- Numerical integration of Newton‘s equation of motion.
- Initial conditions
- Integrator -Velocity Verlet algorithm

from Nongnuch Artrith

Details of the model implemented(contd.)

Preprocessing:

- Cartesian coordinates cannot be used at all,as their numerical values are not invariant with translations and rotations of the system.
- the structural fingerprint for the description of local atomic environment in this project consists of 8 radial and 18 angular Behler-Parrinello basis functions for each combination of atomic species.
- The dimension of the input layer of each atomic NN is 70,as the input nodes of ANN potentials for the energy of oxygen and titanium atoms,respectively are the values of each 8 radial functions for interactions with O and Ti and 18 angular functions for interactions with O-O, O-Ti, Ti-Ti atom pairs ie. $(2 \times 8 + 3 \times 18 = 70)$



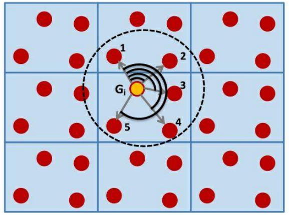
Radial distribution function

The radial function centered at atom i is defined as,

$$G_i^2 = \sum_{j \neq i} e^{-\eta (R_{ij} - R_s)^2} \cdot f_c(R_{ij})$$

The smooth truncation outside cutoff sphere is achieved by cosine cuttoff function,

$$f_c(R_{ij}) = \begin{cases} 0.5\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{for } R_{ij} \leq R_c \\ 0 & \text{for } R_{ij} > R_c \end{cases}$$



Angular distribution function

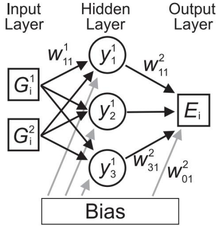
The angular three-body function centered at atom i is given by,

$$G_i^4 = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq ij} (1 + \lambda \cos\theta_{ijk})^{\zeta} \cdot e^{-\eta\left(R_{ij}^2 + R_{ik}^2 + R_{jk}^2\right)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \cdot f_c(R_{jk})$$

from Nongnuch Artrith

Details of the model implemented(contd.)

NN model implementation:



Source:Behler et al

Functionalities implemented	Remarks:
Loading the dataset	
Test-train splitting	99% train,1% test
Min-Max normalization	Range - [-1,1]
Weight-bias initialization	random weights, guessed weights
Data shuffling	
Activation functions	sigmoid,tanh,ReLU
Cost functions	MSE,MAE,RMSE
Forward propagation	
Backward propagation	

Functionalities implemented	Remarks:
Gradient checking	central difference method
NN switcher	
Forward prop through NN assembly	
Backprop through NN assembly	
Optimizers	SGD,SGD momentum,Adam, minibatch GD,RMSprop.
Hyperparameter Tuning	
Correlation coefficient	$r^2 = 1$ for a perfect fit
Saving the trained parameters	(as npz file formats)
Energy Prediction	

Details of the model implemented(contd.)

Optimizers implemented	Update relations:
SGD	$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right)$
SGD with momentum	$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$ $\theta = \theta - v_t$
mini batch GD	$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right)$
RMS-Prop	$E\left[g^2\right]_t = \beta E\left[g^2\right]_{t-1} + (1-\beta) g_t^2$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t$
Adam	$m_t = \beta_1 m_{t-1} + \left(1 - \beta_1\right) g_t$ $v_t = \beta_2 v_{t-1} + \left(1 - \beta_2\right) g_t^2$ $\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$

Preconditioning of NN

The significance of different symmetry functions are balaced by rescaling all of them to a same interval of [-1,1]

$$G_i^{\text{scaled}} = \frac{2\left(G_i - G_{i,min}\right)}{G_{i,\max} - G_{i,\min}} - 1$$

Correlation coefficient:

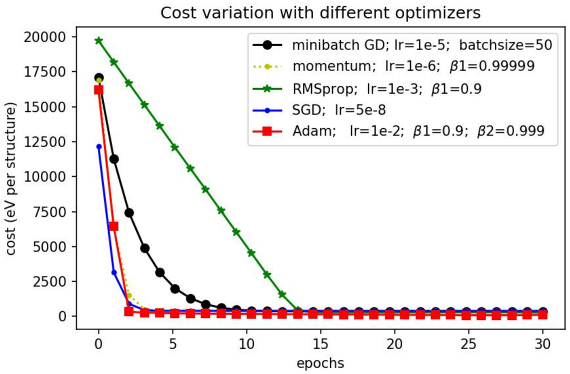
$$r^2 = 1 - \left[\frac{\sum_{i=1}^N \left(E_{i,\text{ref}} - E_{i,\text{NN}}\right)^2}{\left[\sum_{i=1}^N \left(E_{i,\text{ref}} - \left(\frac{1}{N}\sum_{j=1}^N E_{j,\text{ref}}\right)\right)\right]^2}\right]$$

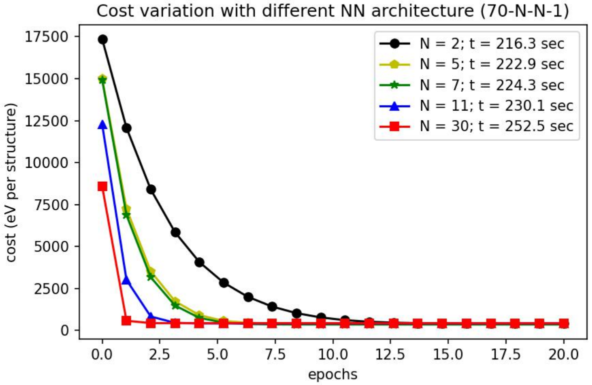
For a perfect fit by the NN model,

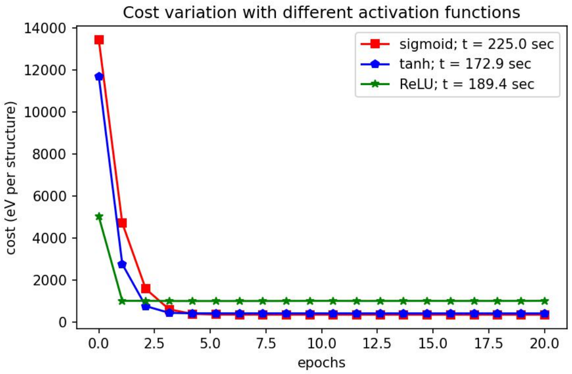
$r^2 = 1$

Details of the model implemented (contd.)

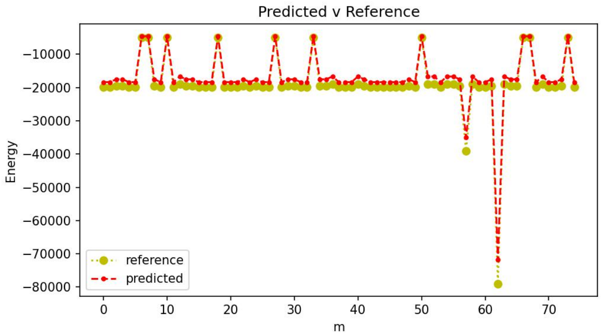
Hyperparameter tuning:

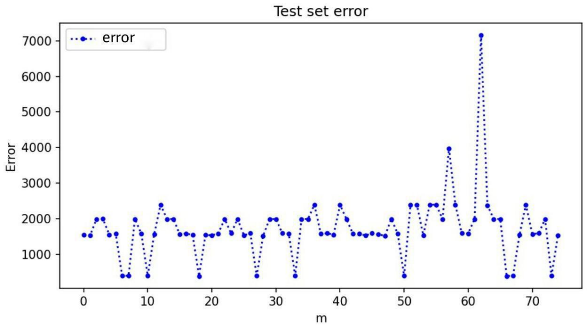

Cost variation with different optimizers

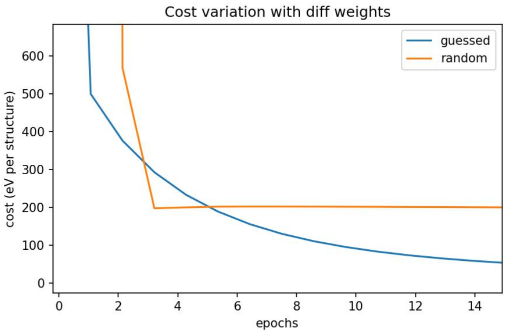

Cost variation with different NN architecture (70-N-N-1)


Cost variation with different activation functions

Local minima issue:


Predicted v Reference


Test set error


Cost variation with diff weights

Details of the model implemented (contd.)

Postprocessing:

Calculation of force:

The forces on all atoms are required in order to numerically integrate the Newton’s laws of motion in the MD simulations.The force component $F_{k,\alpha}$ acting on atom k with respect to coordinate $R_{k,\alpha}$, α = (x, y, z) could be found by applying chain rule,

$$F_{k,\alpha} = -\frac{\partial E}{\partial R_{k,\alpha}} = -\sum_{i=1}^{N} \frac{\partial E_i}{\partial R_{k,\alpha}} = -\sum_{i=1}^{N}\sum_{j=1}^{M_i} \frac{\partial E_i}{\partial G_{i,j}} \frac{\partial G_{i,j}}{\partial R_{k,\alpha}}$$

where N is the number of atoms,M_i is the number of symmetry functions describing atom i.

$\frac{\partial E_i}{\partial G_{i,j}}$ → calculated from the architecture of the neural network model
shape --> (n x 1 x 70)

$\frac{\partial G_{i,j}}{\partial R_{k,\alpha}}$ → found from the definition of employed symmetry functions
shape --> (n x 70 x 3)
(as the gradient raises the tensorial order by 1)

Analytical derivatives of symmetry functions:

The derivative of the cut-off function,

$$\frac{\partial f_c(r)}{\partial r} = -\frac{1}{2}\frac{\pi}{R_c}\sin\left(\frac{\pi r}{R_c}\right)$$

The gradient of the radial symmetry function with respect to the neighbouring atoms j,

$$\vec{\nabla}_j G_i^{\text{radial}} = -\kappa\left(r_{ij}\right)\vec{r}_{ij}$$

where we abbreviated,

$$\kappa(r) = \frac{1}{r}\mathrm{e}^{-\eta(r-rs)^2}\left[\frac{df_c(r)}{dr} - 2\eta\left(r - r_{\mathrm{s}}\right)f_c(r)\right]$$

The gradient of the angular symmetry function with respect to the neighbouring atoms j,

$$\vec{\nabla}_j G_i^{\text{ang}} = 2^{1-\zeta}\sum_{k\neq j,j}\xi^{\text{ang}}\left(\vec{r}_{ij},\vec{r}_{ik}\right)$$
$$\cdot\left[-\left(\phi\left(\vec{r}_{ij},\vec{r}_{ik}\right) - 2\eta + \chi\left(r_{ij}\right)\right)\vec{r}_{ij} + \left(\psi\left(\vec{r}_{ij},\vec{r}_{ik}\right) - 2\eta + \chi\left(r_{jk}\right)\right)\vec{r}_{jk}\right]$$

where,

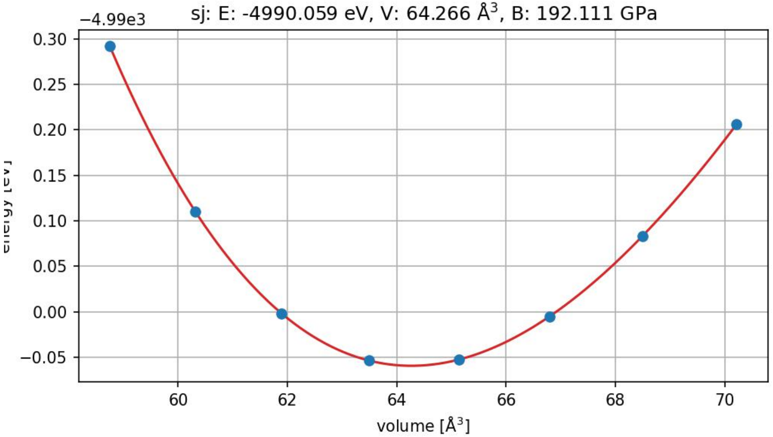
$$\psi\left(\vec{r}_1,\vec{r}_2\right) = -\frac{1}{r_1 r_2}\frac{\lambda\zeta}{1+\lambda\cos\theta\left(\vec{r}_1,\vec{r}_2\right)}$$

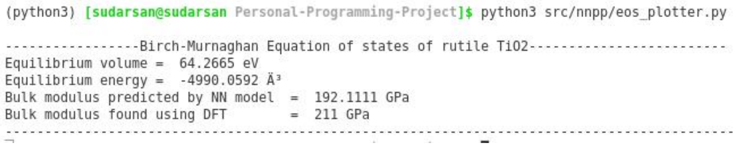
$$\phi\left(\vec{r}_1,\vec{r}_2\right) = -\psi\left(\vec{r}_1,\vec{r}_2\right) - \frac{1}{r_1^2}\frac{\lambda\zeta}{1+\lambda\cos\theta\left(\vec{r}_1,\vec{r}_2\right)}$$

$$\chi(r) = \frac{1}{r f_c(r)}\frac{df_c(r)}{dr}$$

Application of the NN model:

Using the NN model in Equation of states:
The energy and volume of a set of configurations of rutile TiO_2 were used to fit the Birch-Murnaghan equation of states using the ASE package.The energy for each configuration was predicted by the trained NN model in order to find its Bulk modulus.





```
(python3) [sudarsan@sudarsan Personal-Programming-Project]$ python3 src/nnpp/eos_plotter.py

-----------------Birch-Murnaghan Equation of states of rutile TiO2------------------------
Equilibrium volume =  64.2665 eV
Equilibrium energy =  -4990.0592 Å³
Bulk modulus predicted by NN model  =  192.1111 GPa
Bulk modulus found using DFT        =  211 GPa
-----------------------------------------------------------------------------------------
```

Using the NN model in MD simulations:
It is a technique by which the atomic trajectories of a system of N particles can be generated by employing the numerical integration of the Newton's equation of motion.
-Initial condition: The initial positions of the atoms were given from a random dataset which was not used for training the NN model.The atomic velocities were initialized using a Maxwell-Boltzmann distribution,N (0, k_BT /m_i), for a temperature of 1000 K.
-Calculation of force: The force on each atom at each timestep is calculated as the gradient of energy by applying chain rule,where the energy is predicted by the trained NN model.
-Integrator: Velocity-Verlet algorithm was implemented in order to advance the trajectory of atoms over the small time steps.Here we begin with **x**(t_0) and **v**(t_0),then, for atom i,

$$\mathbf{x}_i\left(t_0+\Delta t\right)=\mathbf{x}_i\left(t_0\right)+\mathbf{v}_i\left(t_0\right)\Delta t+\frac{1}{2}\left(\frac{\mathbf{f}_i\left(t_0\right)}{m_i}\right)(\Delta t)^2$$

now we evaluate f(t_0 + Δt), and then,

$$\mathbf{v}_i\left(t_0+\Delta t\right)=\mathbf{v}_i\left(t_0\right)+\frac{1}{2}\left[\frac{\mathbf{f}_i\left(t_0\right)}{m_i}+\frac{\mathbf{f}_i\left(t_0+\Delta t\right)}{m_i}\right]\Delta t$$

where **Δt** is the time increment,$\mathbf{m_i}$ is the mass of atom i, **f** is the force on atom, **x**,the position of atom, **v**,the velocity of atom.

Details on the tests performed

Functionality tested	Number of test cases	Passed?(Yes/No)
Neural network assembly		
Forward propagation	5	Yes
Backward propagation	5	Yes
Overfitting using a single dataset	3	Yes
Training on a small dataset	1	Yes
Sanity check using test set	1	Yes
Unit-tests		
Functions in symmetry routine	15	Yes
Activation functions	18	Yes
Other functions	11	Yes
MD code		
Testing the motion of atoms	2	Yes
Total	61	Yes

Details on the tests performed(contd.)

Forward Propagation: The forward propagation of the NN was tested using known inputs,fixed weights of value 1 and pre-calculated output values.

Backward Propagation: The backward propagation of the NN was tested using a method called **gradient checking**.The numerical gradient was estimated using the method of central differences by introducing small perturbations.The partial derivative of the cost function with respect to each parameter is found by perturbing the parameter forward and backward and using it to find the cost in order to find the numerical gradient using the relation,

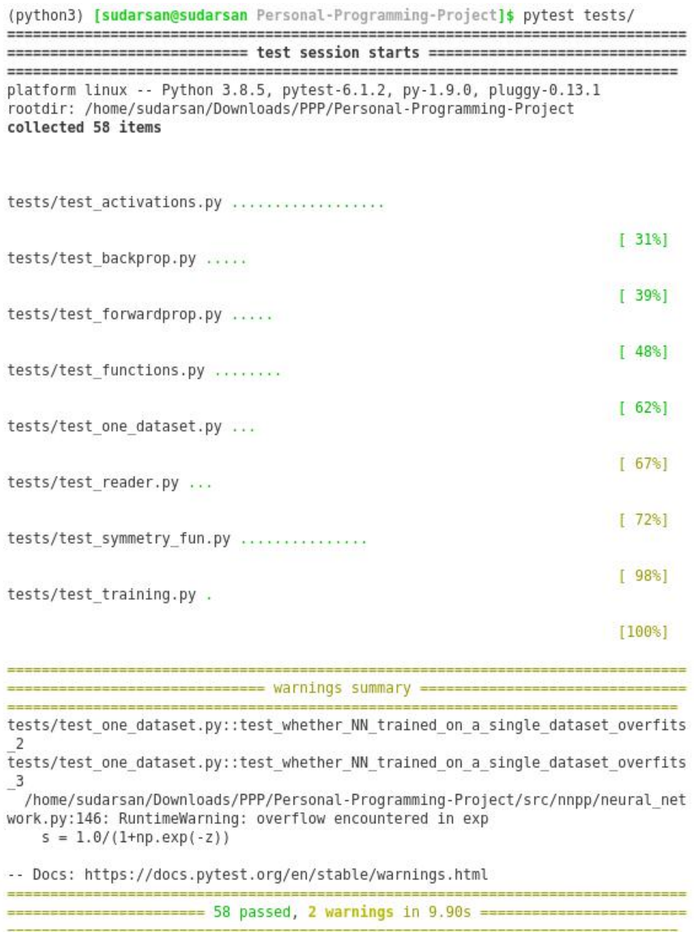
$$\frac{\partial J}{\partial \theta} = \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

Now this numerical gradient is compared with the analytical gradients found using the back-propagation function of the neural network.

Overfitting using a single dataset: A single dataset was used to train the model and after the training,the same dataset was used for the output prediction.It was checked whether the NN was able to over fit the dataset with a 100% training accuracy.

Training on a small dataset: The neural network model was trained using a small dataset to see whether the cost goes below a tolerance value.

Unit testing: Each and every functions used in the neural network package was tested using known inputs and pre-calculated outputs.

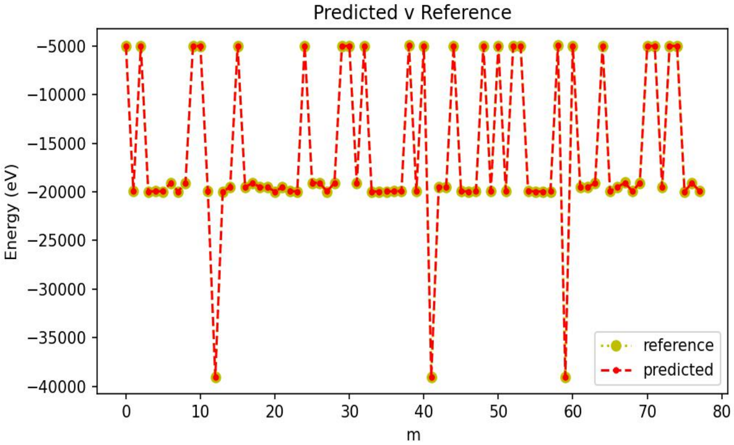


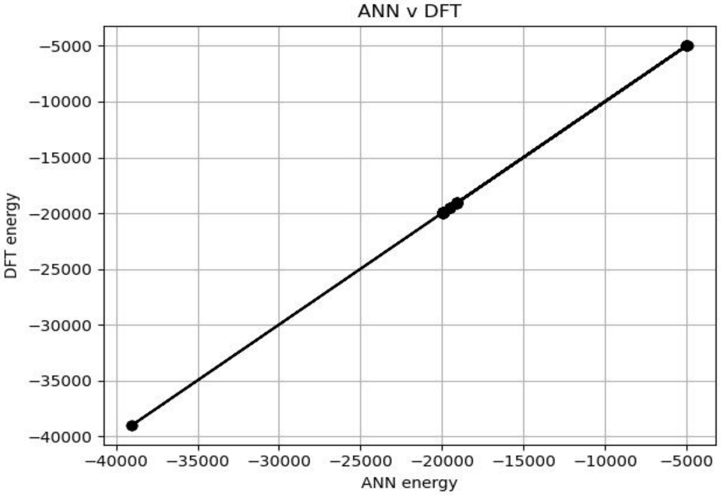
```
(python3) [sudarsan@sudarsan Personal-Programming-Project]$ pytest tests/
================================================================================
========================== test session starts =================================
===============================================================================
platform linux -- Python 3.8.5, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
rootdir: /home/sudarsan/Downloads/PPP/Personal-Programming-Project
collected 58 items


tests/test_activations.py ..................
                                                                        [ 31%]
tests/test_backprop.py .....
                                                                        [ 39%]
tests/test_forwardprop.py .....
                                                                        [ 48%]
tests/test_functions.py ........
                                                                        [ 62%]
tests/test_one_dataset.py ...
                                                                        [ 67%]
tests/test_reader.py ...
                                                                        [ 72%]
tests/test_symmetry_fun.py ...............
                                                                        [ 98%]
tests/test_training.py .
                                                                        [100%]

================================================================================
=========================== warnings summary ===================================
===============================================================================
tests/test_one_dataset.py::test_whether_NN_trained_on_a_single_dataset_overfits
_2
tests/test_one_dataset.py::test_whether_NN_trained_on_a_single_dataset_overfits
_3
  /home/sudarsan/Downloads/PPP/Personal-Programming-Project/src/nnpp/neural_net
work.py:146: RuntimeWarning: overflow encountered in exp
    s = 1.0/(1+np.exp(-z))

-- Docs: https://docs.pytest.org/en/stable/warnings.html
================================================================================
======================= 58 passed, 2 warnings in 9.90s =========================
-------------------------------------------------------------------------------
```

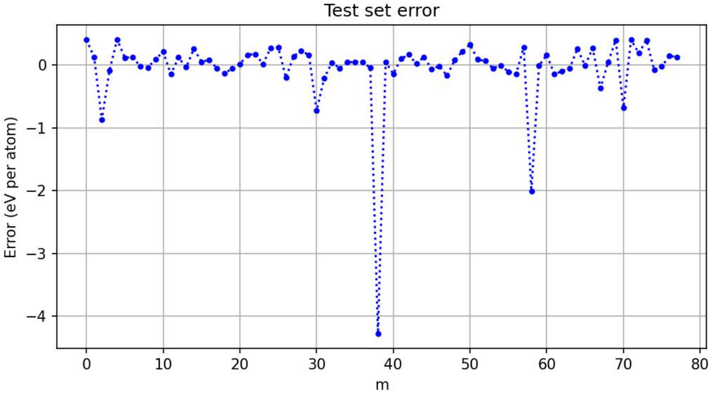
Details on the tests performed (contd.)

Sanity check for the neural network assembly: The trained neural network was used to do predictions on the test data.From the reference dataset,1% was selected randomly to validate the neural network predictions and ensure its transferability.

ANN-DFT Correlation: The correlation between the energy predicted by neural network and reference DFT energy was found by plotting the former on X-axis and the latter on Y-axis.Moreover,the **R^2 values** calculated for both the training set and testing set were found to be 0.99.

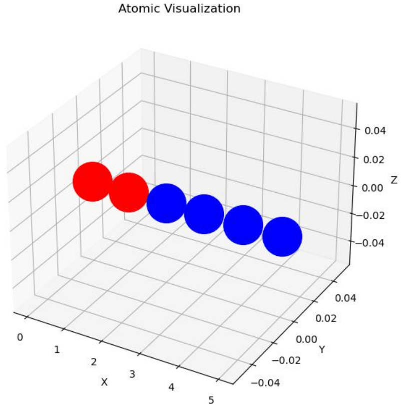

Predicted v Reference


ANN v DFT

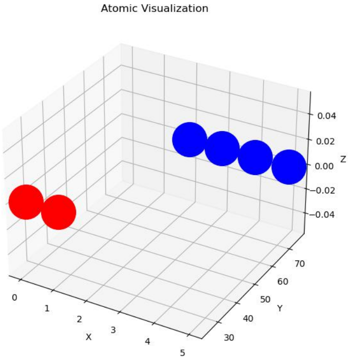

Test set error

Details on the tests performed (contd.)

Test for MD code: 6 atoms were aligned in a line in X direction.Now a constant force was applied in each time step along Y direction.As the force components are in Y directions,motion of the atoms should also should be in Y direction.



(a) before md run



(a) after md run

As the mass of first 2 atoms were greater than the remaining 4,they had a lower acceleration,leading to a varied displacement at the end.

References:

[1] Nongnuch Artrith and Alexander Urban. An implementation of artificial neural-network potentials for atomistic materials simulations:Performance for TiO_2. Computational Materials Science, 114:135–150,2016.

[2] John-Anders Stende. Constructing high-dimensional neural network potentials for molecular dynamics. Master’s thesis, 2017.

[3] Jörg Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. Physical Chemistry Chemical Physics, 13(40):17930–17955, 2011.

[4] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. International Journal of Quantum Chemistry, 115(16):1032–1050, 2015.

[5] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. Physical review letters, 98(14):146401, 2007.

[6] Jörg Behler. Atom-centered symmetry functions for constructing highdimensional neural network potentials. The Journal of chemical physics, 134(7):074106, 2011.

[7] Nongnuch Artrith, Tobias Morawietz, and Jörg Behler. Highdimensional neural-network potentials for multicomponent systems: Applications to zinc oxide. Physical Review B, 83(15):153101, 2011.

[8] Nongnuch Artrith, Björn Hiller, and Jörg Behler. Neural network potentials for metals and oxides – first applications to copper clusters at zinc oxide. physica status solidi (b), 250(6):1191–1203, 2013.

[9] Suresh Kondati Natarajan, Tobias Morawietz, and Jörg Behler. Representing the potential-energy surface of protonated water clusters by high-dimensional neural network potentials. Phys. Chem. Chem. Phys.,17:8356–8371, 2015.

Thank you!