Chapter 6 Learning DeePMD-Kit: A Guide to Building Deep Potential Models

CHAPTER 6

LEARNING DeePMD-KIT: A GUIDE TO BUILDING DEEP POTENTIAL MODELS

Wenshuo Liang, Jinzhe Zeng, Darrin M. York, Linfeng Zhang, and Han Wang

Liang, W., Zeng, J., York, D. M., Zhang, L., and Wang, H., "Learning DeePMD-kit: A guide to building deep potential models," in *A Practical Guide to Recent Advances in Multiscale Modeling and Simulation of Biomolecules*, edited by Y. Wang and R. Zhou (AIP Publishing, Melville, New York, 2023), pp. 6-1–6-20.

6.1 INTRODUCTION

Over the last few decades, molecular dynamics (MD) simulations have attracted a great deal of attention due to their wide range of applications in many fields such as condensed matter physics, materials science, polymer chemistry, and molecular biology. They provide researchers access to examine the behavior of atoms or molecules, which is valuable and has the potential to enrich our knowledge, especially when conducting experiments is difficult, expensive, or even impossible.

It is well recognized that the accuracy of the PES ultimately limits the quality of MD simulations, and accurately representing the PES is a significant challenge in the field of MD simulations. Empirical atomic potential and quantum mechanical (QM) models have long been two commonly used models. The empirical atomic potential models consist of simple low-dimensional terms. They often show excellent computational efficiency but have limited accuracy. The QM models determine the energies and forces on atoms by approximately solving the Schrödinger equation, typically under the Kohn-Sham density functional theory (DFT) formulism (Kohn and Sham, 1965), for electronic structure and exhibit higher accuracy. However, quantum mechanical models are computationally demanding, and they are not reasonably practical for large-scale and/or long-time calculations. Overall, a dilemma exists between the choice of an empirical atomic potential for high efficiency and that of a quantum mechanics model for increased accuracy.

Machine learning (ML) models are emerging as valuable tools to address this dilemma. Descriptors and ML algorithms are two primary components of the current ML models. The former is used to guarantee

https://doi.org/10.1063/9780735425279_006

A Practical Guide to Recent Advances in Multiscale Modeling and Simulation of Biomolecules, Y. Wang and R. Zhou

the system's natural symmetries. The latter is used to establish a direct functional relationship between atomic configurations and potential energy by training on reference data generated by quantum mechanics. Once trained, ML models can provide the same accuracy as the quantum mechanical method used to generate the reference data. Meanwhile, the computational cost scales cubic for DFT with system size, whereas it scales linearly for ML models. So far, different types of ML models have been reported in the literature, such as Behler-Parrinello neural network potentials (BPNNP) (Behler and Parrinello, 2007), Gaussian approximation potentials (GAP) (Bartók *et al.*, 2010), spectral neighbor analysis potentials (SNAP) (Trott *et al.*, 2014), ANI-1 (Smith *et al.*, 2017), SchNet (Schütt *et al.*, 2017), and Deep Potentials (DP) (Han *et al.*, 2018; and Zhang *et al.*, 2018a, 2018b). It is worth pointing out that despite great successes, many challenging issues remain to be tackled (Deringer *et al.*, 2019). For example, neglecting interactions beyond the cut-off radius may lead to systematic prediction errors (Yue *et al.*, 2021).

This chapter focuses on the DP models. In addition to enabling quantum mechanical accuracy, current DP models have the following characteristics: (i) preserving the symmetry of the system, especially when there are multiple elemental species; (ii) having high computational efficiency, being at least five orders of magnitude faster than DFT; (iii) being end-to-end and therefore having little human intervention; (iv) supporting MPI and GPU, making it highly efficient on modern heterogeneous high-performance supercomputers. Thanks to these points, the DP models have been successfully employed in studies of water and water-containing systems (Calegari Andrade *et al.*, 2020; Sommers *et al.*, 2020; Xu *et al.*, 2020; and Tisi *et al.*, 2021), metals and alloys (Zhang *et al.*, 2019; Wang *et al.*, 2020; Jiang *et al.*, 2021; and Wen *et al.*, 2021), phase diagrams (Niu *et al.*, 2020; Yang *et al.*, 2021; and Zhang *et al.*, 2021), high-entropy ceramics (Dai *et al.*, 2020, 2021), chemical reaction (Zeng *et al.*, 2020, 2021a, 2021b), solid-state electrolytes (Huang *et al.*, 2021), ionic liquids (Liang *et al.*, 2021), etc. We refer to Wen *et al.* (2022) for a recent review of DP for material systems. In the following sections, we will introduce the basic theory of the DP method and show how to train a DP model.

6.2 THEORY

Before introducing the DP method, we define the coordinate matrix $\mathcal{R} \in \mathbb{R}^{N\times 3}$ of a system containing N atoms,

$$\mathcal{R} = \left\{\boldsymbol{r}_1^T, \ldots, \boldsymbol{r}_i^T, \ldots, \boldsymbol{r}_N^T\right\}^T, \quad \boldsymbol{r}_i = (x_i, y_i, z_i) \tag{6.1}$$

$\boldsymbol{r}_i$ contains 3 Cartesian coordinates of atom i and $\mathcal{R}$ can be transformed into local environment matrices $\{\mathcal{R}^i\}_{i=1}^N$,

$$\mathcal{R}^i = \left\{\boldsymbol{r}_{1i}^T, \ldots, \boldsymbol{r}_{ji}^T, \ldots, \boldsymbol{r}_{N_i,i}^T\right\}^T, \quad \boldsymbol{r}_{ji} = (x_{ji}, y_{ji}, z_{ji}) \tag{6.2}$$

where j and N_i are indexes of the neighbors of atom i within the cut-off radius r_c, and $\boldsymbol{r}_{ji} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$ is defined as the relative coordinate.

In the DP method, a system's total energy E is constructed as the sum of atomic energies.

$$E = \sum_i E_i \tag{6.3}$$

with E_i being the local atomic energy of the atom i. E_i depends on the local environment of the atom i:

$$E = \sum_i E_i = \sum_i E(\mathcal{R}^i) \tag{6.4}$$

The mapping of $\mathcal{R}^i$ to E_i is constructed in two steps. First, each $\mathcal{R}^i$ is mapped to a feature matrix, also called the descriptor, $\mathcal{D}^i$ to preserve the translational, rotational, and permutational symmetries of the system. As shown in Fig. 6.1, $\mathcal{R}^i \in \mathbb{R}^{N_i \times 3}$ is first transformed into generalized local environment matrix $\tilde{\mathcal{R}}^i \in \mathbb{R}^{N_i \times 4}$.

$$\{x_{ji}, y_{ji}, z_{ji}\} \mapsto \{s(r_{ji}), \hat{x}_{ji}, \hat{y}_{ji}, \hat{z}_{ji}\} \tag{6.5}$$

where $r_{ji} = \| \boldsymbol{r}_{ji} \|$, $\hat{x}_{ji} = \frac{s(r_{ji}) x_{ji}}{r_{ji}}$, $\hat{y}_{ji} = \frac{s(r_{ji}) y_{ji}}{r_{ji}}$, and $z_{ji} = \frac{s(r_{ji}) z_{ji}}{r_{ji}}$. $s(r_{ji})$ is a weighting function to reduce the weight of particles that are more distant from the atom i, defined as

$$s(r_{ji}) = \begin{cases} \dfrac{1}{r_{ji}}, & r_{ji} < r_{cs} \\ \dfrac{1}{r_{ji}} \left\{ \left(\dfrac{r_{ji} - r_{cs}}{r_c - r_{cs}} \right)^3 \left(-6 \left(\dfrac{r_{ji} - r_{cs}}{r_c - r_{cs}} \right)^2 + 15 \dfrac{r_{ji} - r_{cs}}{r_c - r_{cs}} - 10 \right) + 1 \right\} & r_{cs} < r_{ji} < r_c \\ 0, & r_{ji} > r_c \end{cases}. \tag{6.6}$$

Here, r_{ji} is the Euclidean distance between atoms i and j, and r_{cs} is the smooth cut-off parameter. By introducing $s(r_{ji})$ the components in the $\tilde{\mathcal{R}}^i$ smoothly go to zero from r_{cs} to r_c. Then, $s(r_{ji})$, i.e., the first column of $\tilde{\mathcal{R}}^i$, is mapped to an embedding matrix $\mathcal{G}^{i1} \in \mathbb{R}^{N_i \times M_1}$ with M_1 outputs, through an embedding neural network $\mathcal{N}^e$, i.e., $\mathcal{G}^{i1} = \mathcal{N}^e(s(r_{ji}))$. By taking the first $M_2 (< M_1)$ columns of $\mathcal{G}^{i1}$, we obtain another embedding matrix $\mathcal{G}^{i2} \in \mathbb{R}^{N_i \times M_2}$. Finally, we define the feature matrix $\mathcal{D}^i \in \mathbb{R}^{M_1 \times M_2}$ of atom i:

$$\mathcal{D}^i = (\mathcal{G}^{i1})^T \tilde{\mathcal{R}}^i (\tilde{\mathcal{R}}^i)^T \mathcal{G}^{i2} \tag{6.7}$$

In Eq. (6.7), translational and rotational symmetries are preserved by the matrix product $\tilde{\mathcal{R}}^i (\tilde{\mathcal{R}}^i)^T$, and permutational symmetry is preserved by the matrix product $(\mathcal{G}^i)^T \tilde{\mathcal{R}}^i$. Next, each $\mathcal{D}^i$ is mapped to a local atomic energy E_i through a fitting network $\mathcal{N}^f$, i.e., $E_i = \mathcal{N}^f(\mathcal{D}^i)$.

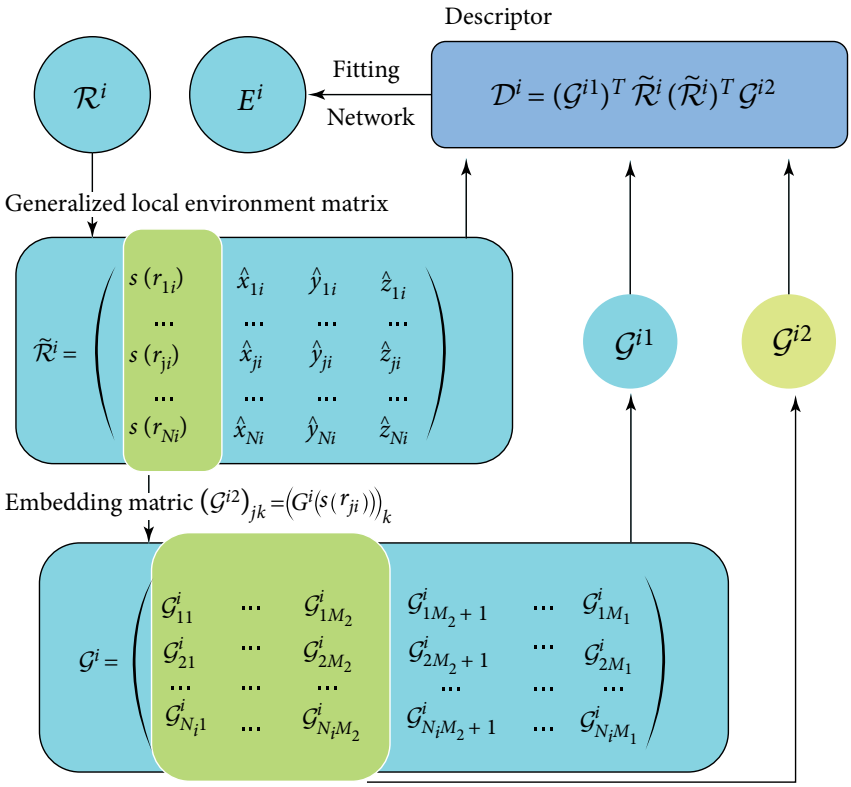

FIG. 6.1
Graphic illustration of the mapping from the local environment of local atomic energy. Figure adapted from Zhang *et al., Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18* (Association for Computing Machinery, 2018b), pp. 4441–4451. Copyright 2018b Association for Computing Machinery.

Both the embedding network $\mathcal{N}^e$ and the fitting network $\mathcal{N}^f$ are feed-forward neural networks containing several hidden layers. The mapping from input data d_l^{in} of the previous layer to output data d_k^{out} of the next layer is composed of a linear and a non-linear transformation.

$$d_k^{\text{out}} = \varphi\left(\sum_{kl} w_{kl} d_l^{\text{in}} + b_k\right) \tag{6.8}$$

In Eq. (6.8), w_{kl} is the connecting weight, b_k is the bias weight, and φ is a non-linear activation function. It needs to be noted that only linear transformations are applied at the output nodes. The parameters

contained in the embedding and fitting networks, i.e., w_{kl} and b_k, are obtained by minimizing the loss function L:

$$L(p_\varepsilon, p_f, p_\xi) = \frac{p_\varepsilon}{N}\Delta\varepsilon^2 + \frac{p_f}{3N}\sum_i |\Delta \boldsymbol{F}_i|^2 + \frac{p_\xi}{9N}||\Delta\xi||^2 \tag{6.9}$$

where $\Delta\varepsilon$, ΔF_i, and $\Delta\xi$ denote root mean square error (RMSE) in energy, force, and virial, respectively. During the training process, the prefactors p_ε, p_f, and p_ξ are determined by

$$p(t) = p^{\text{limit}}\left[1 - \frac{r_l(t)}{r_l^0}\right] + p^{\text{start}}\left[\frac{r_l(t)}{r_l^0}\right] \tag{6.10}$$

where $r_l(t)$ and r_l^0 are the learning rates at training step t and training step 0. $r_l(t)$ is defined as

$$r_l(t) = r_l^0 \times d_r^{t/d_s} \tag{6.11}$$

where d_r and d_s are the decay rate and decay steps, respectively. The decay rate d_r is required to be less than 1. The reader is referred to the original papers (Han *et al.*, 2018; and Zhang *et al.*, 2018a, 2018b) for more details.

6.3 MATERIALS

The DP model is generated using the DeePMD-kit package (v2.1.0) (Wang *et al.*, 2018). The training data are converted into the format of DeePMD-kit using a tool named dpdata (v0.2.5). We noted that dpdata only works with Python 3.5 and later versions. The MD simulations are carried out using LAMMPS (29 Sep 2021) (Plimpton, 1995) integrated with DeePMD-kit. Details of dpdata and DeePMD-kit installation and execution can be found in the DeepModeling official GitHub site https://github.com/deepmodeling/. OVITO (Stukowski, 2010) is used for the visualization of the MD trajectory. All input files can be found at https://github.com/deepmodeling-activity/AIP-chapter-tutorial.

6.4 METHODS

In this section, we will introduce to the readers the primary usage of the DeePMD-kit, taking a gas-phase methane molecule as an example. Typically, the DeePMD-kit workflow contains data preparation, training/freezing/compressing/testing and performing molecular dynamics. We notice that here the compressing step has not been commonly adopted by most ML-based physical models, but it deserves to consider this 10-minute step after training since the compressed DP model typical speeds up DP-based calculations by an order of magnitude, and consumes an order of magnitude less memory. More information can be found in Lu *et al.* (2022).

6.4.1 Data preparation

The training data contains the atom type, the simulation box, the atom coordinate, the atom force, the system energy, and the virial. A snapshot of a molecular system having this information is called a frame. A data system includes many frames with the same number of atoms and atom types. For example, a molecular dynamics trajectory can be converted into a data system, with each time step corresponding to a frame in the system. The DeePMD-kit adopts a compressed data format. All training data should first be converted into this format and then used by the DeePMD-kit.

6.4.2 Training

6.4.2.1 Preparing training data

As an example, go to the data folder:

```
$ cd 00.data
$ ls
OUTCAR
```

The OUTCAR file was produced by an *ab initio* molecular dynamics (AIMD) simulation of a gas-phase methane molecule using VASP (Garcia-Viloca *et al.*, 2004; and Giese *et al.*, 2021). Now start an interactive python environment, for example,

```
$ python
```

then execute the following commands:

```
import dpdata
import numpy as np
data = dpdata.LabeledSystem('OUTCAR', fmt='vasp/outcar')
print('# the data contains %d frames' % len(data))
```

On the screen, you can see that the OUTCAR file contains 200 frames of data. We randomly pick 40 frames as validation data and the rest as training data. The parameter set_size specifies the set size.

```
index_validation = np.random.choice(200,size=40,replace= False)
index_training = list(set(range(200))-set(index_validation))
data_training = data.sub_system(index_training)
data_validation = data.sub_system(index_validation)
data_training.to_deepmd_npy('training_data')
data_validation.to_deepmd_npy('validation_data')
print('# the training data contains %d frames' % len(data_training))
print('# the validation data contains %d frames' % len(data_validation))
```

The commands import a data system from the OUTCAR (with format vasp/outcar) and then dump it into the compressed format (NumPy compressed arrays). The data in DeePMD-kit format are stored in folder 00.data.

```
$ ls training_data
set.000 type.raw type_map.raw
$ cat training_data/type.raw
0 0 0 1
```

Since all frames in the system have the same atom types and atom numbers, we only need to specify the type information once for the whole system,

```
$ cat training_data/type_map.raw
H C
```

where element H is given type 0, and element C is given type 1.

6.4.2.2 Preparing input script

Once the data preparation is done, we can proceed with training. Now go to the training directory,

```
$ cd ../01.train
$ ls
input.json
```

where `input.json` gives you an example training script. The options are explained in detail in the DeePMD-kit manual, so they are not comprehensively explained here. The model has two subsections, `descriptor` and `fitting_net`, in which the parameters of embedding and fitting networks are specified, respectively.

```
"model":{
  "type_map": ["H", "C"],
       "descriptor":{
                "type": "se_e2_a",
                "rcut": 6.00,
                "rcut_smth": 0.5,
                "sel": [4, 1],
                "neuron": [10, 20, 40],
                "resnet_dt": false,
                "axis_neuron": 4,
                "seed": 1,
                "_comment": "that's all"
        },
```

```
        "fitting_net":{
                "neuron": [100, 100, 100],
                "resnet_dt": true,
                "seed": 1,
                "_comment": "that's all"
        },
        "_comment": "that's all"
},
```

The `se_e2_a` descriptor is the one introduced in this book chapter. The item neurons set the size of the embedding and fitting network to `[10, 20, 40]` and `[100, 100, 100]`, respectively. The components in $\tilde{\mathcal{R}}^i$ to smoothly go to zero from 0.5 to 6 Å.

The following are the parameters that specify the learning rate and loss function.

```
"learning_rate":{
        "type": "exp",
        "decay_steps": 5000,
        "start_lr": 0.001,
        "stop_lr": 3.51e-8,
        "_comment": "that's all"
},
        "loss":{
        "type": "ener",
        "start_pref_e": 0.02,
        "limit_pref_e": 1,
        "start_pref_f": 1000,
        "limit_pref_f": 1,
        "start_pref_v": 0,
        "limit_pref_v": 0,
        "_comment": "that's all"
},
```

In the loss function (Eq. 6.9), p_ε increases from 0.02 to 1 eV^{-2}, and p_f decreases from 1000 to 1 $\text{Å}^2\ \text{eV}^{-2}$ progressively, which means that the force term dominates at the beginning, while energy and virial terms become important at the end. This strategy is very effective and reduces the total training time. p_ξ is set to 0 eV^{-2}, indicating that no virial data are included in the training process.

The starting learning rate, stop learning rate, and decay steps are set to 0.001, 3.51e-8, and 5000, respectively. The model is trained for 10^6 steps. According to Eq. (6.10), we can derive the decay rate value as 0.95.

The training parameters are given in the following:

```
  "training":{
     "training_data":{
       "systems": ["../00.data/training_data/"],
       "batch_size": "auto",
       "_comment": "that's all"
},
   "validation_data":{
       "systems": ["../00.data/validation_data/"],
       "batch_size": "auto",
       "numb_btch": 1,
       "_comment": "that's all"
},
   "numb_steps": 1000000,
       "seed": 10,
       "disp_file": "lcurve.out",
       "disp_freq": 1000,
       "save_freq": 10000,
},
```

6.4.2.3 Training a model

After the training script is prepared, we can start the training process with the DeePMD-kit by simply running

```
$ dp train input.json
```

On the screen, you see the information of the data system(s)

```
-----------------------------------------------
---Summary of DataSystem: training ------------
found 1 system(s):
       system natoms bch_sz n_bch prob pbc
../00.data/training_data/ 5 7 22 1.000 T
-----------------------------------------------
---Summary of DataSystem: validation ----------
found 1 system(s):
      system natoms bch_sz n_bch prob pbc
../00.data/validation_data/ 5 7 5 1.000 T
```

and the starting and final learning rate of this training

```
start training at lr 1.00e-03 (== 1.00e-03), decay_step
    5000, decay_rate 0.950006, final lr will be 3.51e-08
```

If everything works fine, you will see on the screen information printed every 1000 steps, like

```
batch 1000 training time 7.61 s, testing time 0.01 s
batch 2000 training time 6.46 s, testing time 0.01 s
batch 3000 training time 6.50 s, testing time 0.01 s
batch 4000 training time 6.44 s, testing time 0.01 s
batch 5000 training time 6.49 s, testing time 0.01 s
batch 6000 training time 6.46 s, testing time 0.01 s
batch 7000 training time 6.24 s, testing time 0.01 s
batch 8000 training time 6.39 s, testing time 0.01 s
batch 9000 training time 6.72 s, testing time 0.01 s
batch 10000 training time 6.41 s, testing time 0.01 s
saved checkpoint model.ckpt
```

At the end of the 10000th batch, the model is saved in Tensorflow's checkpoint file model.ckpt. At the same time, the training and testing errors are presented in file lcurve.out.

```
$ head -n 2 lcurve.out
#step rmse_val rmse_trn rmse_e_val rmse_e_trn rmse_f_val rmse_f_trn lr
0 1.34e+01 1.47e+01 7.05e-01 7.05e-01 4.22e-01 4.65e -01 1.00e-03
```

and

```
$ tail -n 2 lcurve.out
999000 1.24e-01 1.12e-01 5.93e-04 8.15e-04 1.22e-01 1.10e-01 3.7e-08
1000000 1.31e-01 1.04e-01 3.52e-04 7.74e-04 1.29e-01 1.02e-01 3.5e-08
```

Columns 4, 5 and 6, 7 present energy and force training and testing errors, respectively. After 10^6 steps of training, the energy testing error is less than 1 meV/atom and the force testing error is around 120 meV/Å. It is also observed that the force testing error is systematically (but slightly) larger than the training error, which implies a slight over-fitting to the rather small dataset. See Fig. 6.2 for the visual illustration.

When the training process is stopped abnormally, we can restart the training from the provided checkpoint by simply running

```
$ dp train --restart model.ckpt input.json
```

In the lcurve.out file, you can see the training and testing errors, like

```
538000 3.12e-01 2.16e-01 6.84e-04 7.52e-04 1.38e-01 9.52e-02 4.1e-06
539000 3.37e-01 2.61e-01 7.08e-04 3.38e-04 1.49e-01 1.15e-01 4.1e-06
#step rmse_val rmse_trn rmse_e_val rmse_e_trn rmse_f_val rmse_f_trn lr
530000 2.89e-01 2.15e-01 6.36e-04 5.18e-04 1.25e-01 9.31e-02 4.4e-06
531000 3.46e-01 3.26e-01 4.62e-04 6.73e-04 1.49e-01 1.41e-01 4.4e-06
```

Note that input.json needs to be consistent with the previous one.

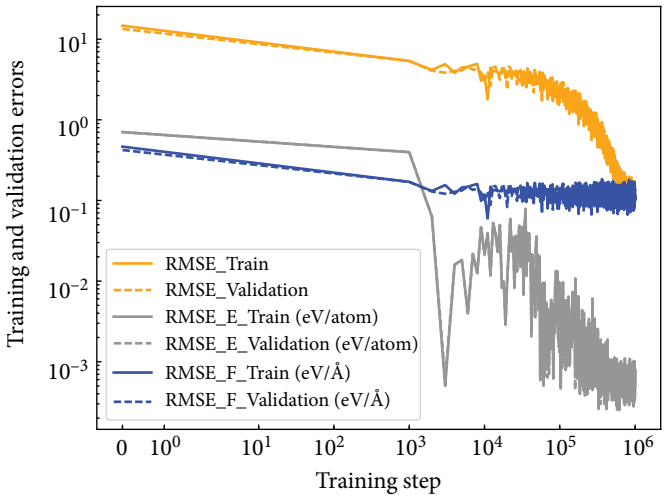

FIG. 6.2
The training and validation errors of the DP model: the square root of loss for training (solid yellow) and validation (dashed yellow), root mean square errors (RMSEs) in force for training (solid blue) and validation (dashed blue), and RMSEs in energy for training (solid grey) and validation (dashed grey). The RMSEs in the energy are normalized by the number of atoms in the system.

6.4.2.4 Freezing and compressing a model

At the end of the training process, the model parameters saved in TensorFlow's checkpoint file should be frozen as a model file that is usually ended with the extension.pb. Simply execute

```
$ dp freeze -o graph.pb
```

On the screen, you will see information printed, like

```
Restoring parameters from ./model.ckpt-1000000
1264 ops in the final graph
```

and it will output a model file named graph.pb in the current directory.

The graph.pb file can be compressed in the following way:

```
$ dp compress -i graph.pb -o graph-compress.pb
```

On the screen, you will see information printed, like

```
stage 1: compress the model
built lr
built network
```

```
built training
initialize model from scratch
finished compressing

stage 2: freeze the model
Restoring parameters from model-compression/model.ckpt
840 ops in the final graph
```

and it will output a model file named graph-compress.pb.

6.4.2.5 Testing a model

We can check the quality of the trained model by running

```
$ dp test -m graph-compress.pb -s ../00.data/
     validation_data -n 40 -d results
```

On the screen you see the information of the prediction errors of validation data

```
# number of test data : 40
Energy RMSE: 3.168050e-03 eV
Energy RMSE/Natoms : 6.336099e-04 eV
Force RMSE: 1.267645e-01 eV/A
Virial RMSE: 2.494163e-01 eV
Virial RMSE/Natoms: 4.988326e-02 eV
# ----------------------------------------
```

and it will output files named `results.e.out` and `results.f.out` in the current directory. See Fig. 6.3 for the visual illustration.

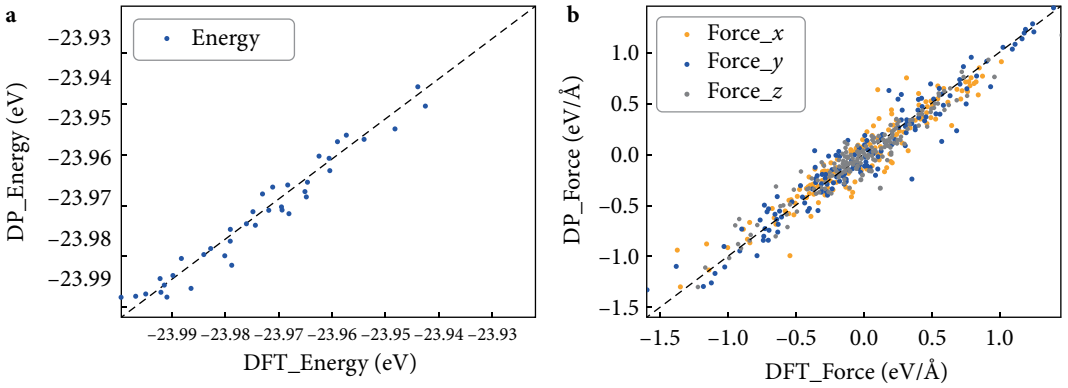

FIG. 6.3
Comparisons of DFT and DP predicted energies (a) and forces (b) on 40 frames of a gas-phase methane molecule.

6.4.3 Run MD with LAMMPS

Now let's switch to the lammps directory to check the necessary input files for running DeePMD with LAMMPS.

```
$ cd ../02.lmp
```

Firstly, we soft-link the output model in the training directory to the current directory

```
$ ln -s ../01.train/graph-compress.pb
```

Then, we have three files

```
$ ls
conf.lmp graph-compress.pb in.lammps
```

where conf.lmp gives the initial configuration of a gas-phase methane MD simulation, and the file in.lammps is the lammps input script. One may check in.lammps and find that it is a rather standard LAMMPS input file for a MD simulation, with only two exception lines:

```
pair_style graph-compress.pb
pair_coeff **
```

where the pair style `deepmd` is invoked and the model file `graph-compress.pb` is provided, which means that the atomic interaction will be computed by the DP model that is stored in the file `graph-compress.pb`.

One may execute LAMMPS the standard way

```
$ lmp -i in.lammps
```

After waiting for a while, the MD simulation finishes, and the log.lammps and ch4.dump files are generated. They store thermodynamic information and the trajectory of the molecule, respectively. One may want to visualize the trajectory by, e.g. OVITO

```
$ ovito ch4.dump
```

to check the evolution of the molecular configuration.

In this case, the performance of training models and MD simulations is listed in Table 6.1.

Table 6.1
Performance of training and MD simulations using different hardware (unit: ms/step).[a]

	Training		MD simulations	
Hardware	**FP64**	**FP32**	**FP64**	**FP32**
AMD EPYC 7742 (32 cores)	5.10	3.87	0.615	0.506
NVIDIA GeForce RTX 3080 Ti	7.74	6.84	1.38	1.01
NVIDIA GeForce RTX 3090	13.39	12.54	3.80	4.12

[a] Both training and MD simulations were conducted following the example given in the previous sections. FP64 means that the NNs in the model use the double-precision floating-point format, while FP32 means that the NNs in the model use the single-precision floating-point format.

6.5 BIOCATALYSIS SIMULATIONS WITH AMBER AND DeePMD-kit

In this section, we briefly introduce some recent progress on biocatalysis simulations that require a more delicate treatment on the DeePMD-kit. We refer to Zeng *et al.* (2021a) for more details.

Simulations of biomolecular processes present special challenges. These processes are carried out by large biological macromolecules (e.g., proteins and/or nucleic acids) in environments that may consist of water molecules, ions, lipids, and small molecules. This translates into a highly complex heterogeneous condensed phase environment, where intermolecular forces are governed by a delicate balance of electrostatic, hydrophobic, and hydrogen bonding interactions mediates by the aqueous environment and ion atmosphere.

In this type of environment, biological catalysts are able to accelerate the rates of chemical reactions necessary to carry out life processes under ambient conditions with unequivocal efficiency—in some cases up to 19 orders of magnitude (Garcia-Viloca *et al.*, 2004)! In order to achieve these remarkable rate enhancements, protein and nucleic acid enzymes must fold into 3-dimensional structures able to selectively bind substrates and create active sites that tune electrostatic interactions and position chemical functional groups and bound ions to participate in catalysis. While the reactive residues within the active site may only contain up to a few hundred atoms (and require a full quantum mechanical treatment), the equally critical-surrounding environment may consist of tens to hundreds of thousands of atoms. Taken together, this presents not only considerable challenges with respect to the efficient evaluation of the energy and forces but also introduces new challenges with respect to the requirement to appropriately sample the necessary degrees of freedom in order to determine the reaction-free energy surface (Giese *et al.*, 2021) from which mechanistic pathways can be determined.

In order to tackle these biological problems, recently, the DeePMD-kit was integrated with the AMBER molecular simulation software suite (Case *et al.*, 2020) in order to perform biocatalysis simulations (Zeng *et al.*, 2021a). This interface enables *ab initio* combined quantum mechanical/molecular mechanical (QM/MM) simulations with a rigorous treatment of long-ranged electrostatic interactions under periodic boundary conditions (Giese and York, 2016; and Pan *et al.*, 2021), and affords a powerful tool to gain insight into the pathways of biocatalysis reactions, their transition states and intermediates, and environmental factors that modulate reactivity (Gaines *et al.*, 2019; and Ganguly *et al.*, 2020). In order to achieve high (*ab initio* level) accuracy at low computational cost, a new deep potential range correction (DPRc) has been developed that enables short-ranged QM/MM interactions to be tuned for higher accuracy, and the correction smoothly vanishes within a specified cutoff such that it can be easily integrated with molecular mechanical (MM) force fields that utilize a non-bonded list such as those in AMBER. An active learning training procedure has been developed and validated (Zeng

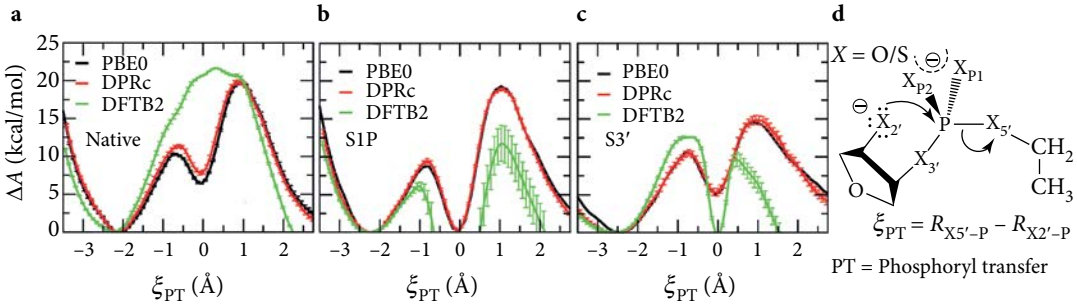

FIG. 6.4
Results for DPRc model trained to *ab initio* DFT (PBE0/6-31G*) QM/MM data (Giese *et al.*, 2022) for non-enzymatic reactions of a native system (all oxygen) and variants with thio substitutions at XP1 and X3′ positions.

et al., 2021a), and very recently applied to develop DPRc potentials that closely reproduce *ab initio* QM/MM-free energy profiles (Giese *et al.*, 2022).

Mechanistic pathways from QM/MM + DPRc simulations provide important insights that can be further validated experimentally (Figs. 6.4 and 6.5).

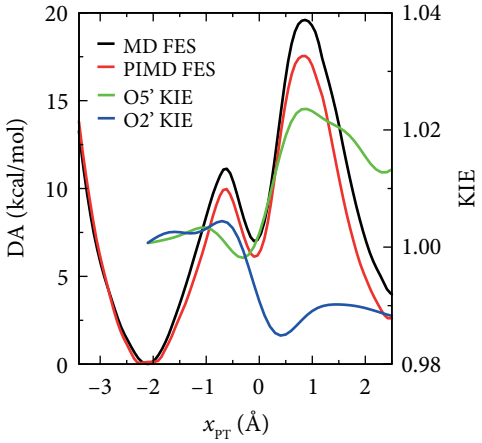

FIG. 6.5
QM/MM + DPRc FES and O2′ and O5′ isotope effects (Giese *et al.*, 2022) as a function of the reaction coordinate for the native reaction in Fig. 6.4(d). MD and PIMD QM/MM simulations were performed with SPC/Fw (Wu *et al.*, 2006) and q-SPC/Fw (Paesani *et al.*, 2006) water models, respectively.

6.6 NOTES

1. **Deep Potential-Smooth Edition (DeepPot-SE) model.** In this chapter, the DP model introduced is the smooth edition. The original DP model (Han *et al.*, 2018; and Zhang *et al.*, 2018a) satisfies the symmetries of the system by establishing a local coordinate system for each atom and its neighbor in the cut-off radius r_c, which introduces discontinuities in the model. This has negligible influence on a trajectory from canonical sampling but might not be sufficient for calculations of dynamical and mechanical properties.
2. **Cut-off radius and deep potential long-range (DPLR) model.** For covalent and metal-based materials, a cut-off radius of 6 Å is recommended. For systems where long-range interactions are important, such as ionic solids, careful testing is required before determining the cut-off radius. In addition, long-range interactions can be included by training the DPLR model (Zhang *et al.*, 2022).
3. **Generating training data with concurrent learning.** In Sec. 6.4.1, the training data are converted from the trajectory generated by the AIMD simulation. However, neighboring configurations in the trajectory are often highly correlated, resulting in redundant training data. A concurrent learning platform named Deep Potential GENerator (DP-GEN) has been developed to avoid this issue. The reader is referred to the original papers of DP-GEN for details (Zhang *et al.*, 2020).
4. **Train an atomic dipole or polarizability model.** In Sec. 6.4.2, we trained an energy model (potential energy surface) for a gas-phase methane molecule using energy and force data. The DeePMD-kit also allows training a model based on other physical quantities, such as the dipole moment or polarizability. A detailed explanation of how to train an atomic dipole or polarizability model for a water system is provided in the DeePMD-kit's documentation (https://docs.deepmodeling.org/projects/deepmd/en/v2.1.0). The training data and input scripts for the water system can be found in the /examples/water_tensor/ folder of the DeePMD-kit's source code.

ACKNOWLEDGMENTS

J.Z. and D.M.Y. are grateful for financial support provided by the National Institutes of Health (Grant No. GM107485 to D.M.Y.). Computational resources were provided by the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant ACI-1548562.56 (specifically, the resources EXPANSE at SDSC through allocation TG-CHE190067). The work of H.W. was supported by the National Science Foundation of China under Grant Nos. 11871110 and 12122103.

REFERENCES

Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G., "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," Phys. Rev. Lett. **104**, 136403 (2010).

Behler, J. and Parrinello, M., "Generalized neural-network representation of high-dimensional potential-energy surfaces," Phys. Rev. Lett. **98**, 146401 (2007).

Calegari Andrade, M. F., Ko, H. Y., Zhang, L., Carref, R., and Selloni, A., "Free energy of proton transfer at the water–TiO_2 interface from ab initio deep potential molecular dynamics," Chem. Sci. **11**, 2335–2341 (2020).

Case, D. A., Belfon, K., Ben-Shalom, I. Y., Brozell, S. R., Cerutti, D. S., Cheatham, III T. E. *et al.*, *AMBER 20* (University of California, San, Francisco, CA, 2020).

Dai, F. Z., Sun, Y., Wen, B., Xiang, H., and Zhou, Y., "Temperature dependent thermal and elastic properties of high entropy $(Ti_{0.2}Zr_{0.2}Hf_{0.2}Nb_{0.2}Ta_{0.2})B_2$: Molecular dynamics simulation by deep learning potential," J. Mater. Sci. Technol. **72**, 8–15 (2021).

Dai, F. Z., Wen, B., Sun, Y., Xiang, H., and Zhou, Y., "Theoretical prediction on thermal and mechanical properties of high entropy $(Zr_{0.2}Hf_{0.2}Ti_{0.2}Nb_{0.2}Ta_{0.2})C$ by deep learning potential," J. Mater. Sci. Technol. **43**, 168–174 (2020).

Deringer, V. L., Caro, M. A., and Csányi, G., "Machine learning interatomic potentials as emerging tools for materials science," Adv. Mater. **31**, e1902765 (2019).

Gaines, C. S., Giese, T. J., and York, D. M., "Cleaning up mechanistic debris generated by twister ribozymes using computational RNA enzymology," ACS Catal. **9**(7), 5803–5815 (2019).

Ganguly, A., Weissman, B. P., Giese, T. J., Li, N. S., Hoshika, S., Saieesh, R. *et al.*, "Confluence of theory and experiment reveals the catalytic mechanism of the Varkud satellite ribozyme," Nat. Chem. **12**, 193–201 (2020).

Garcia-Viloca, M., Gao, J., Karplus, M., and Truhlar, D. G., "How enzymes work: Analysis by modern rate theory and computer simulations," Science **303**, 186–195 (2004).

Giese, T. J., Ekesan, Ş, and York, D. M., "Extension of the variational free energy profile and multistate Bennett acceptance ratio methods for high-dimensional potential of mean force profile analysis," J. Phys. Chem. A **125**, 4216–4232 (2021).

Giese, T. J. and York, D. M., "Ambient-potential composite Ewald method for ab initio quantum mechanical/molecular mechanical molecular dynamics simulation," J. Chem. Theory Comput. **12**, 2611–2632 (2016).

Giese, T. J., Zeng, J., Ekesan, Ş, and York, D. M., "Combined QM/MM, machine learning path integral approach to compute free energy profiles and kinetic isotope effects in RNA cleavage reactions," J. Chem. Theory Comput. **18**, 4304–4317 (2022).

Han, J., Zhang, L., Car, R., and Weinan, E., "Deep potential: A general representation of a many-body potential energy surface," CiCP **23**(3), 629–639 (2018).

Huang, J., Zhang, L., Wang, H., Zhao, J., Cheng, J., and Weinan, E., "Deep potential generation scheme and simulation protocol for the $Li_{10}GeP_2S_{12}$-type superionic conductors," J. Chem. Phys. **154**, 094703 (2021).

Jiang, W., Zhang, Y., Zhang, L., and Wang, H., "Accurate deep potential model for the Al–Cu–Mg alloy in the full concentration space*," Chinese Phys. B **30**, 050706 (2021).

Kohn, W. and Sham, L. J., "Self-consistent equations including exchange and correlation effects," Phys. Rev. **140**(4A), A1133–A1138 (1965).

Liang, W., Lu, G., and Yu, J., "Machine-learning-driven simulations on microstructure and thermophysical properties of $MgCl_2$–KCl eutectic," ACS Appl. Mater. Interfaces **13**, 4034–4042 (2021).

Lu, D., Jiang, W., Chen, Y., Zhang, L., Jia, W., Wang, H. *et al.*, "DP compress: A model compression scheme for generating efficient deep potential models," J. Chem. Theory Comput. **18**(9), 5559–5567 (2022).

Niu, H., Bonati, L., Piaggi, P. M., and Parrinello, M., "Ab initio phase diagram and nucleation of gallium," Nat. Commun. **11**, 2654 (2020).

Paesani, F., Zhang, W., Case, D. A., Cheatham III T. E., and Voth, G. A., "An accurate and simple quantum model for liquid water," J. Chem. Phys. **125**, 184507 (2006).

Pan, X., Nam, K., Epifanovsky, E., Simmonett, A. C., Rosta, E., and Shao, Y., "A simplified charge projection scheme for long-range electrostatics in ab initio QM/MM calculations," J. Chem. Phys. **154**, 024115 (2021).

Plimpton, S., "Fast parallel algorithms for short-range molecular dynamics," J. Comput. Phys. **117**(1), 1–19 (1995).

Schütt, K. T., Kindermans, P. J., Sauceda, H. E., Chmiela, S., Tkatchenko, A., and Müller, K. R., "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions," in Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17 (Association for Computing Machinery, 2017), pp. 992–1002.

Smith, J. S., Isayev, O., and Roitberg, A. E., "ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost," Chem. Sci. **8**(4), 3192–3203 (2017).

Sommers, G. M., Calegari Andrade, M. F., Zhang, L., Wang, H., and Car, R., "Raman spectrum and polarizability of liquid water from deep neural networks," Phys. Chem. Chem. Phys. **22**(19), 10592–10602 (2020).

Stukowski, A., "Visualization and analysis of atomistic simulation data with OVITO–the open visualization tool," Model. Simul Mater Sci. Eng. **18**, 015012 (2010).

Tisi, D., Zhang, L., Bertossa, R., Wang, H., Car, R., and Baroni, S., "Heat transport in liquid water from first-principles and deep neural network simulations," Phys. Rev. B **104**, 224202 (2021).

Trott, C. R., Hammond, S. D., and Thompson, A. P., "SNAP: Strong scaling high fidelity molecular dynamics simulations on leadership-class computing platforms," in *Lecture Notes in Computer Science* (Springer, 2014), Vol. 8488, pp. 19–34.

Wang, H., Zhang, L., Han, J., and Weinan, E., "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics," Comput. Phys. Commun. **228**, 178–184 (2018).

Wang, H., Zhang, Y., Zhang, L., and Wang, H., "Crystal structure prediction of binary alloys via deep potential," Front. Chem. **8**, 589795 (2020).

Wen, T., Wang, R., Zhu, L., Zhang, L., Wang, H., Srolovitz, D. J. *et al.*, "Specialising neural network potentials for accurate properties and application to the mechanical response of titanium," npj Comput Mater. **7**, 206 (2021).

Wen, T., Zhang, L., Wang, H., Weinan, E., and Srolovitz, D. J., "Deep potentials for materials science," Mater. Futures **1**(2), 022601 (2022).

Wu, Y., Tepper, H. L., and Voth, G. A., "Flexible simple point-charge water model with improved liquid-state properties," J. Chem. Phys. **124**, 024503 (2006).

Xu, J., Zhang, C., Zhang, L., Chen, M., Santra, B., and Wu, X., "Isotope effects in molecular structures and electronic properties of liquid water via deep potential molecular dynamics based on the SCAN functional," Phys. Rev. B **102**(21), 214113 (2020).

Yang, M., Karmakar, T., and Parrinello, M., "Liquid-liquid critical point in phosphorus," Phys. Rev. Lett. **127**, 080603 (2021).

Yue, S., Muniz, M. C., Calegari Andrade, M. F., Zhang, L., Car, R., and Panagiotopoulos, A. Z., "When do short-range atomistic machine-learning models fall short?," J. Chem. Phys. **154**, 034111 (2021).

Zeng, J., Cao, L., Xu, M., Zhu, T., and Zhang, J. Z. H., "Complex reaction processes in combustion unraveled by neural network- based molecular dynamics simulation," Nat. Commun. **11**, 5713 (2020).

Zeng, J., Giese, T. J., Ekesan, Ş, and York, D. M., "Development of range-corrected deep learning potentials for fast, accurate quantum mechanical/molecular mechanical simulations of chemical reactions in solution," J. Chem. Theory Comput. **17**, 6993–7009 (2021a).

Zeng, J., Zhang, L., Wang, H., and Zhu, T., "Exploring the chemical space of linear alkane pyrolysis via deep potential generator," Energy Fuels **35**, 762–769 (2021b).

Zhang, L., Han, J., Wang, H., Car, R., and Weinan, E., "Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics," Phys. Rev. Lett. **120**, 143001 (2018a).

Zhang, L., Han, J., Wang, H., Saidi, W. A., Car, R., and Weinan, E., "End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems," in Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18 (Association for Computing Machinery, 2018b), pp. 4441–4451.

Zhang, L., Lin, D. Y., Wang, H., Car, R., and Weinan, E., "Active learning of uniformly accurate interatomic potentials for materials simulation," Phys. Rev. Mater. **3**, 023804 (2019).

Zhang, L., Wang, H., Car, R., and Weinan, E., "Phase diagram of a deep potential water model," Phys. Rev. Lett. **126**, 236001 (2021).

Zhang, Y., Wang, H., Chen, W., Zeng, J., Zhang, L., Wang, H. *et al.*, "DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models," Comput. Phys. Commun. **253**, 107206 (2020).

Zhang, L., Wang, H., Muniz, M. C., Panagiotopoulos, A. Z., Car, R., and Weinan, E., "A deep potential model with long-range electrostatic interactions," J. Chem. Phys. **156**, 124107 (2022).