DEEPMD-KIT训练与MD模拟

2020/09/03

软件环境

- 目前DeePMD-kit软件及对应的软件环境已在210.34.15.205和121.192.191.51节点上安装
- 具体的使用指导和规范请参考：
  - https://chenggroup.github.io/wiki/gpu_usage
  - https://chenggroup.github.io/wiki/softwares_usage/DeePMD-kit
- 以下功能演示环节将使用205节点进行

以下演示环节

- 请登陆测试账号：ssh tutorial@210.34.15.205
- 密码：6z34lGYkbUU
- 登陆后请创建自己的文件夹，并执行:

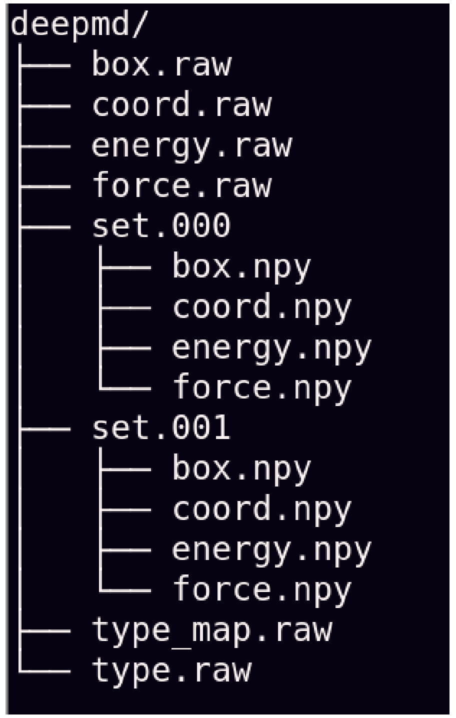
```
git clone https://github.com/chenggroup/new-comer-tutorial.git
```

数据准备

DeePMD-kit中数据的单位

Property	Unit
Time	ps
Length	Å
Energy	eV
Force	eV/Å
Pressure	Bar

■训练数据集的结构



```
deepmd/
├── box.raw
├── coord.raw
├── energy.raw
├── force.raw
├── set.000
│   ├── box.npy
│   ├── coord.npy
│   ├── energy.npy
│   └── force.npy
├── set.001
│   ├── box.npy
│   ├── coord.npy
│   ├── energy.npy
│   └── force.npy
├── type_map.raw
└── type.raw
```

假设体系是一个甲烷分子（CH_4）体系，
原始数据是一个3个离子步的MD轨迹，
因而数据包含3帧，每帧包含4个原子。

box.npy	3x9（每帧包含一个三维盒子）
coord.npy	3x15（每个原子有xyz坐标）
force.npy	3x15（每个原子的受力有xyz分量）
energy.npy	3x1（每帧包含一个能量数值）
virial.npy	3x9（可以没有，每帧包含一个3x3的张量）
type.raw	5x1（元素种类为2）
type_map.raw	2x1（非必须，type的对应关系）

dpdata

- Deepmodeling团队提供的结构信息处理工具，可以对常见的结构文件如POSCAR、OUTCAR、xyz、lammpstrj等提供支持。
- 使用参考：https://github.com/deepmodeling/dpdata
- OUTCAR/CP2K单点能数据->deepmd_npy：可以使用dpdata提供的to_deepmd_npy

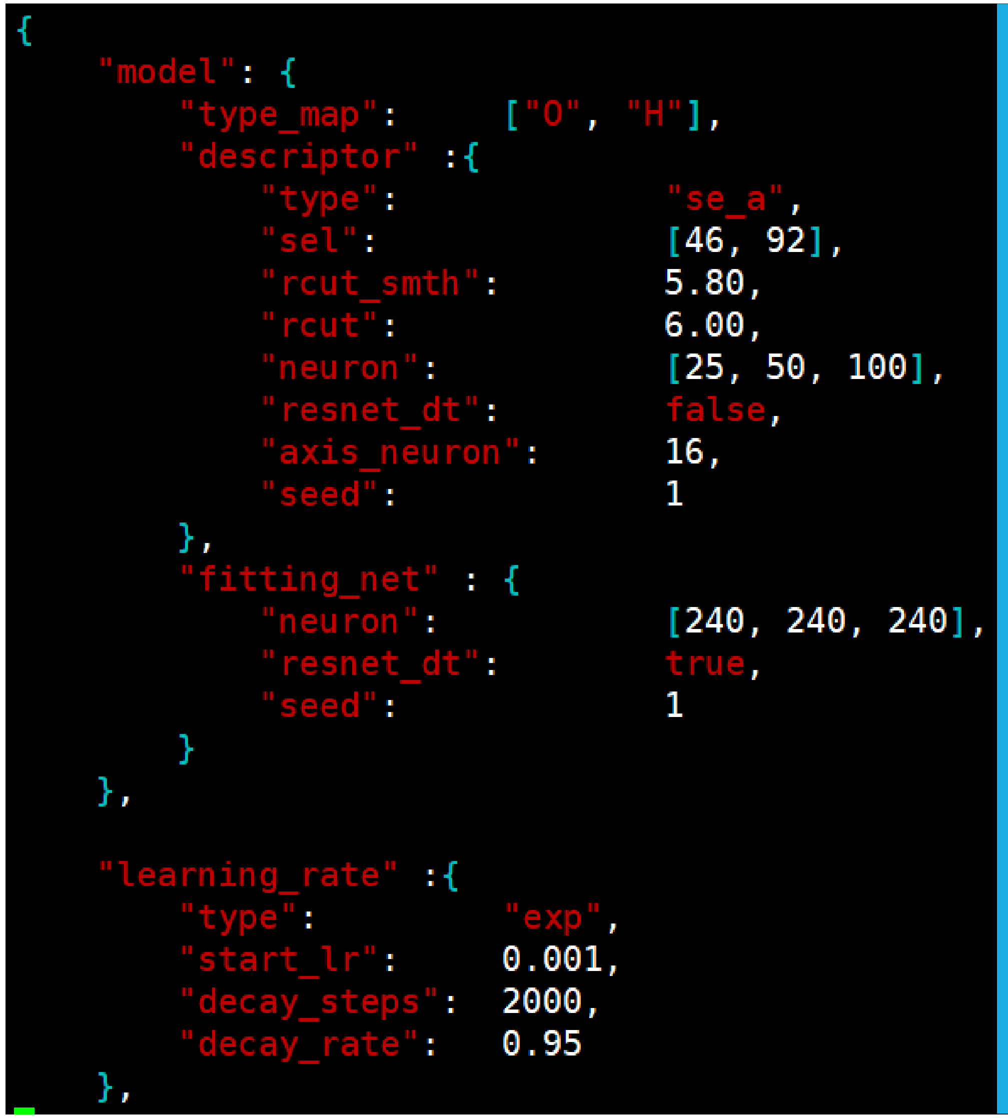
```python
from dpdata import LabeledSystem,MultiSystems
from glob import glob
"""
process multi systems
"""
fs=glob('./*/OUTCAR’)
# remember to change here !!!
ms=MultiSystems()
for f in fs:
    try:
        ls=LabeledSystem(f)
    except:
        print(f)
    if len(ls)>0:
        ms.append(ls)

ms.to_deepmd_raw('deepmd')
ms.to_deepmd_npy('deepmd')
```

数据转换策略

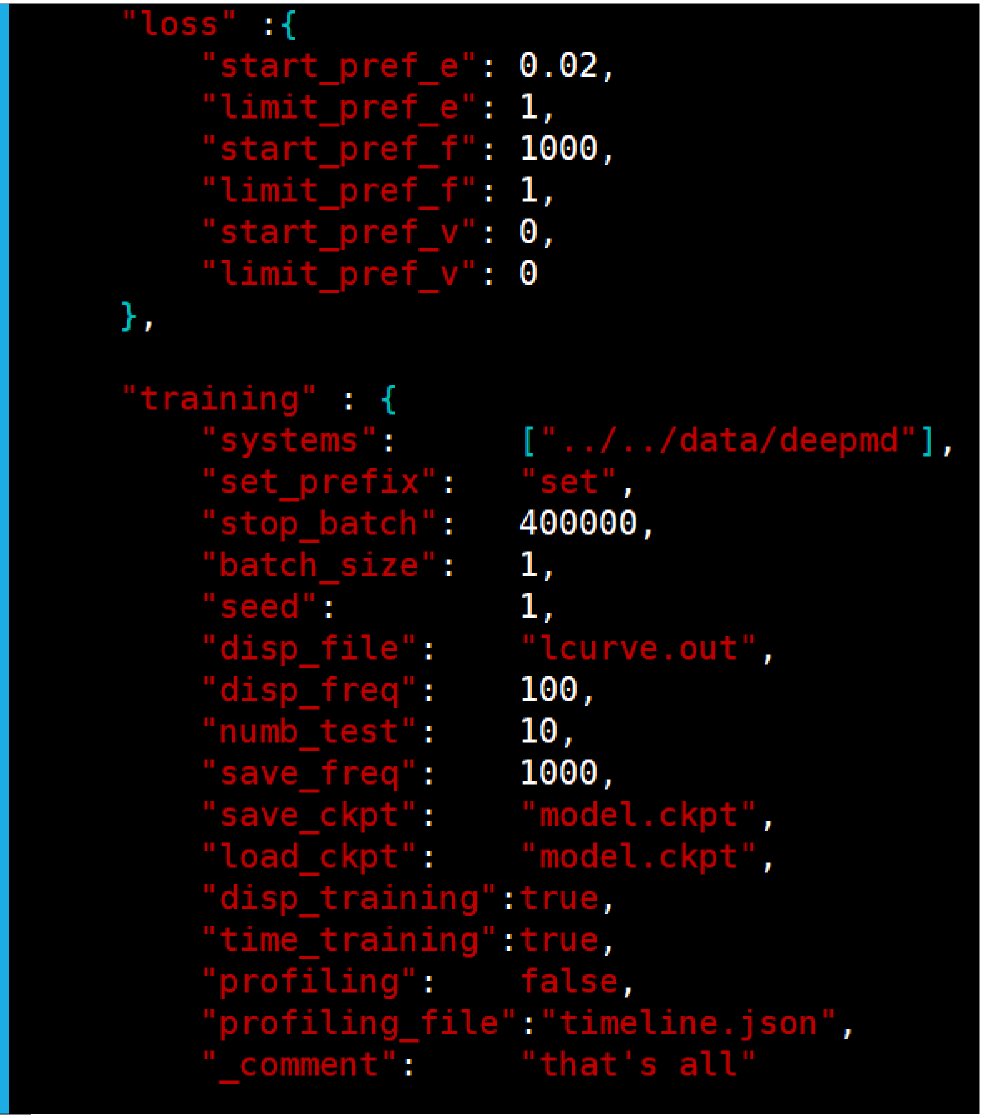
- 其他结构/输出数据：可以使用ASE等软件读取数据，通过numpy转换成对应的矩阵元
- 注意不同数据集元素及其对应的type索引应保持一致。

```
import numpy as np
from glob import glob
from ase.io import read

stcs_list = glob('*/OUTCAR')
stcs_list.sort()
stcs = [read(stc) for stc in stc_list]
os.makedirs('/some/place/data/set.000', exist_ok=True)
pos = np.array([i.get_positions().reshape(-1) for i in stcs])
frc = np.array([i.get_forces().reshape(-1) for i in stcs])
ener = np.array([i.get_potential_energy() for i in stcs])
box = np.array([i.cell.reshape(-1) for i in stcs])
t_map = stcs[0].get_chemical_symbols()
ele = {'Cu': 0, 'C': 1,'O': 2,}
t = np.array([ ele[i] for i in type_map])
np.save('/some/place/data/set.000/coord.npy', pos)
np.save('/some/place/data/set.000/box.npy', box)
np.save('/some/place/data/set.000/energy.npy', ener)
np.save('/some/place/data/set.000/force.npy', frc)
np.savetxt('/some/place/data/type_map.raw', t_map)
np.savetxt('/some/place/data/type.raw', t)
```

参数设置——input.json



```
{
    "model": {
        "type_map":     ["O", "H"],
        "descriptor" :{
            "type":             "se_a",
            "sel":              [46, 92],
            "rcut_smth":        5.80,
            "rcut":             6.00,
            "neuron":           [25, 50, 100],
            "resnet_dt":        false,
            "axis_neuron":      16,
            "seed":             1
        },
        "fitting_net" : {
            "neuron":           [240, 240, 240],
            "resnet_dt":        true,
            "seed":             1
        }
    },

    "learning_rate" :{
        "type":         "exp",
        "start_lr":     0.001,
        "decay_steps":  2000,
        "decay_rate":   0.95
    },
```



```
    "loss" :{
        "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
    },

    "training" : {
        "systems":      ["../../data/deepmd"],
        "set_prefix":   "set",
        "stop_batch":   400000,
        "batch_size":   1,
        "seed":         1,
        "disp_file":    "lcurve.out",
        "disp_freq":    100,
        "numb_test":    10,
        "save_freq":    1000,
        "save_ckpt":    "model.ckpt",
        "load_ckpt":    "model.ckpt",
        "disp_training":true,
        "time_training":true,
        "profiling":    false,
        "profiling_file":"timeline.json",
        "_comment":     "that's all"
```

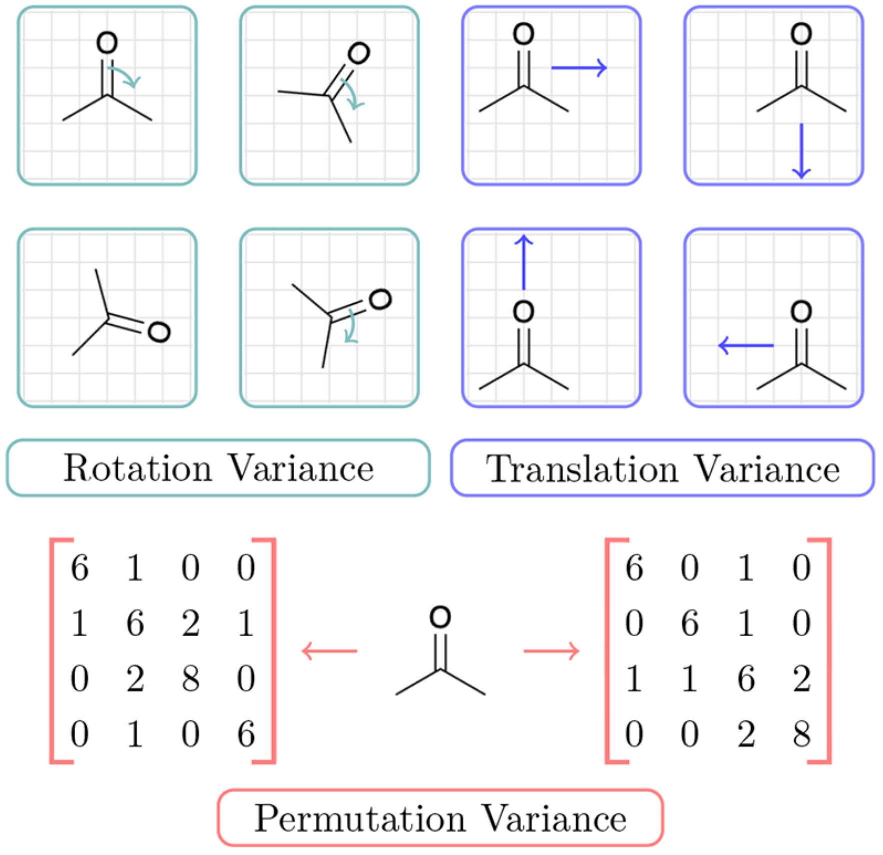
三种不变性

- 描述化学环境与能量对应关系的描述符需要满足三种不变性。

旋转不变

平移不变

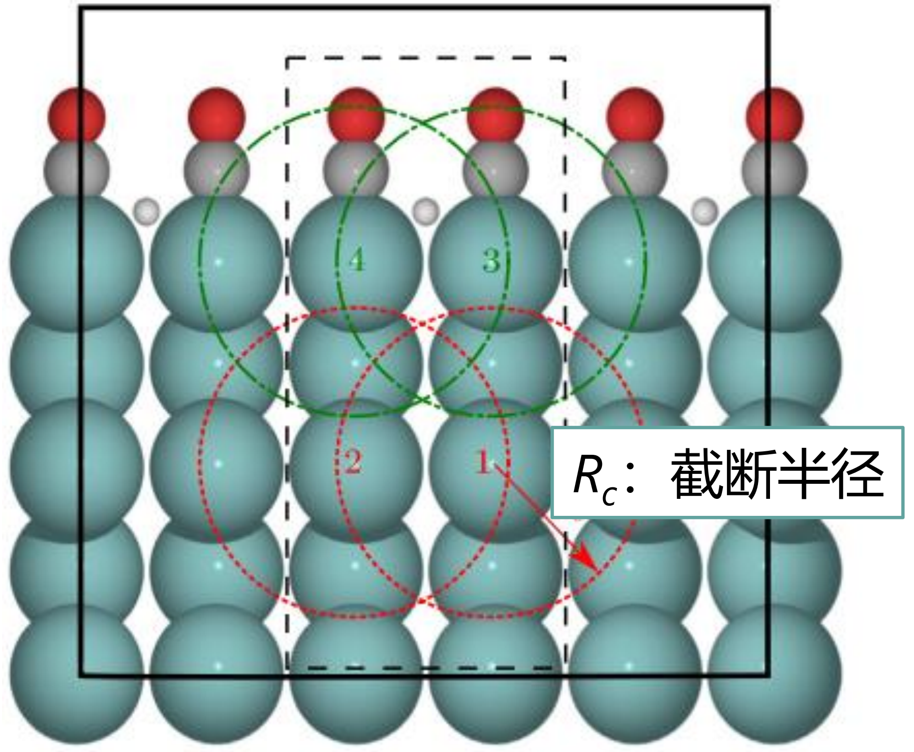
排列不变



$$\begin{bmatrix} 6 & 1 & 0 & 0 \\ 1 & 6 & 2 & 1 \\ 0 & 2 & 8 & 0 \\ 0 & 1 & 0 & 6 \end{bmatrix} \qquad \begin{bmatrix} 6 & 0 & 1 & 0 \\ 0 & 6 & 1 & 0 \\ 1 & 1 & 6 & 2 \\ 0 & 0 & 2 & 8 \end{bmatrix}$$

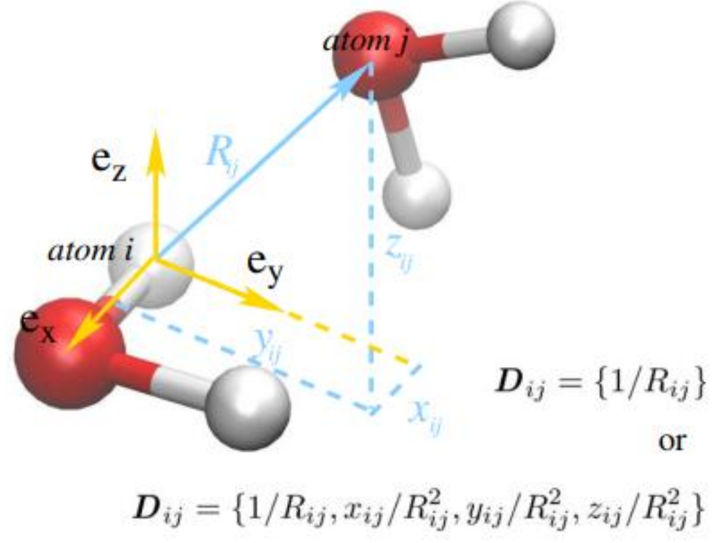
原子邻域环境

Local approximation $E = \sum_i E_i$



R_c：截断半径

对截断半径内的原子环境建立局域坐标，并映射到总能量在该原子上的分量



$\mathbf{D}_{ij} = \{1/R_{ij}\}$

or

$\mathbf{D}_{ij} = \{1/R_{ij}, x_{ij}/R_{ij}^2, y_{ij}/R_{ij}^2, z_{ij}/R_{ij}^2\}$

$$\left\{(\tilde{\mathbf{R}}_1, E_1), (\tilde{\mathbf{R}}_2, E_2), \ldots\right\} \xrightarrow{\text{feature map and regression}} E = \sum_{i=1}^{N} \hat{E}_i(\mathbf{G}_i(\tilde{\mathbf{R}}^{(\mathrm{loc})}))$$

深度势能平滑模型（DeePMD-SE）

- i 原子周围截断半径内的局域环境笛卡尔坐标表示

$$\mathcal{R}^i = \{\boldsymbol{r}_{1i}^T, \cdots, \boldsymbol{r}_{ji}^T, \cdots, \boldsymbol{r}_{N_i,i}^T\}^T,\ \boldsymbol{r}_{ji} = (x_{ji}, y_{ji}, z_{ji})$$

- 定义一般坐标表示：

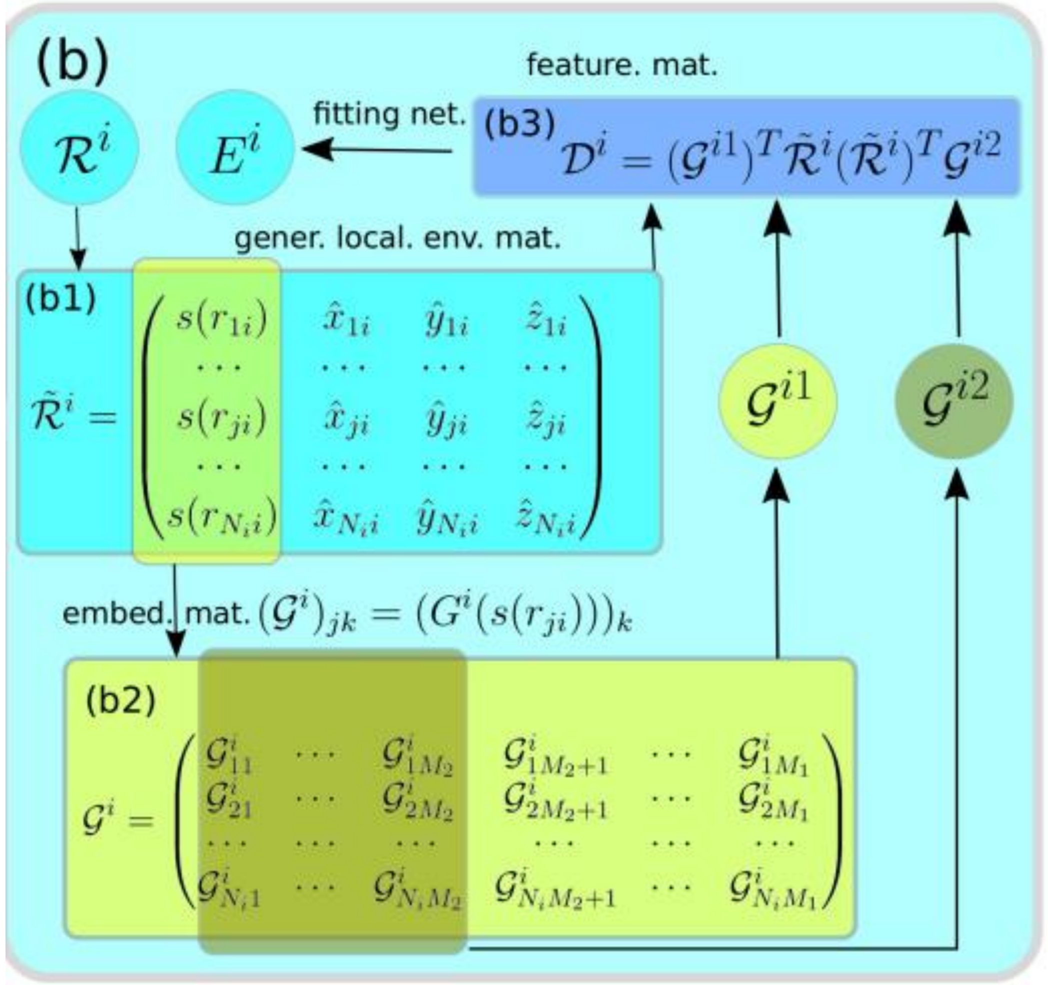
$$\{x_{ji}, y_{ji}, z_{ji}\} \mapsto \{s(r_{ji}), \hat{x}_{ji}, \hat{y}_{ji}, \hat{z}_{ji}\}$$

$$\hat{x}_{ji} = \frac{s(r_{ji})x_{ji}}{r_{ji}},\ \hat{y}_{ji} = \frac{s(r_{ji})y_{ji}}{r_{ji}},\ \hat{z}_{ji} = \frac{s(r_{ji})z_{ji}}{r_{ji}}$$

$$s(r_{ji}) = \begin{cases} \dfrac{1}{r_{ji}}, & r_{ji} < r_{cs}. \\ \dfrac{1}{r_{ji}}\left\{\dfrac{1}{2}\cos\left[\pi\dfrac{(r_{ji}-r_{cs})}{(r_c-r_{cs})}\right]+\dfrac{1}{2}\right\}, & r_{cs} < r_{ji} < r_c. \\ 0, & r_{ji} > r_c. \end{cases}$$

- 平移旋转不变: $\tilde{\mathcal{R}}^i\ (\tilde{\mathcal{R}}^i)^T$

- 排列不变： $\left(\mathcal{G}^{i1}\right)^T\tilde{\mathcal{R}}^i = \sum_j \boldsymbol{G}(\boldsymbol{r}_{ij})\,\tilde{\boldsymbol{r}}(\boldsymbol{r}_{ij})$



(b)
feature. mat.
fitting net.
(b3) $\mathcal{D}^i = (\mathcal{G}^{i1})^T\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T\mathcal{G}^{i2}$
$\mathcal{R}^i$
E^i
gener. local. env. mat.
(b1)
$\tilde{\mathcal{R}}^i = \begin{pmatrix} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i}) & \hat{x}_{N_i} & \hat{y}_{N_i} & \hat{z}_{N_i} \end{pmatrix}$
$\mathcal{G}^{i1}$
$\mathcal{G}^{i2}$
embed. mat. $(\mathcal{G}^i)_{jk} = (G^i(s(r_{ji})))_k$
(b2)
$\mathcal{G}^i = \begin{pmatrix} \mathcal{G}^i_{11} & \cdots & \mathcal{G}^i_{1M_2} & \mathcal{G}^i_{1M_2+1} & \cdots & \mathcal{G}^i_{1M_1} \\ \mathcal{G}^i_{21} & \cdots & \mathcal{G}^i_{2M_2} & \mathcal{G}^i_{2M_2+1} & \cdots & \mathcal{G}^i_{2M_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{G}^i_{N_i1} & \cdots & \mathcal{G}^i_{N_iM_2} & \mathcal{G}^i_{N_iM_2+1} & \cdots & \mathcal{G}^i_{N_iM_1} \end{pmatrix}$

训练过程

能量：$E = \sum_i \mathcal{N}_{\alpha_i}\left(\mathcal{D}_{\alpha_i}(r_i, \{r_j\}_{j\in n(i)})\right)$

力： $\boldsymbol{F}_i = -\nabla_{r_i} E$

损失函数：

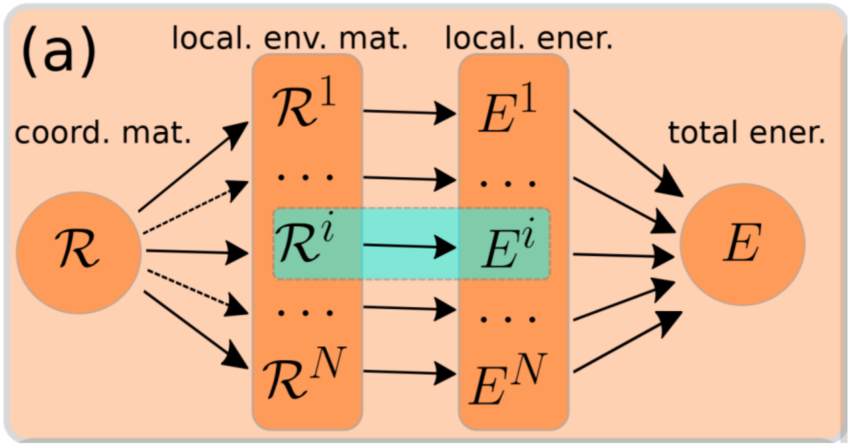
$$p(t) = p^{\text{limit}}\left[1 - \frac{r_l(t)}{r_l^0}\right] + p^{\text{start}}\left[\frac{r_l(t)}{r_l^0}\right],$$

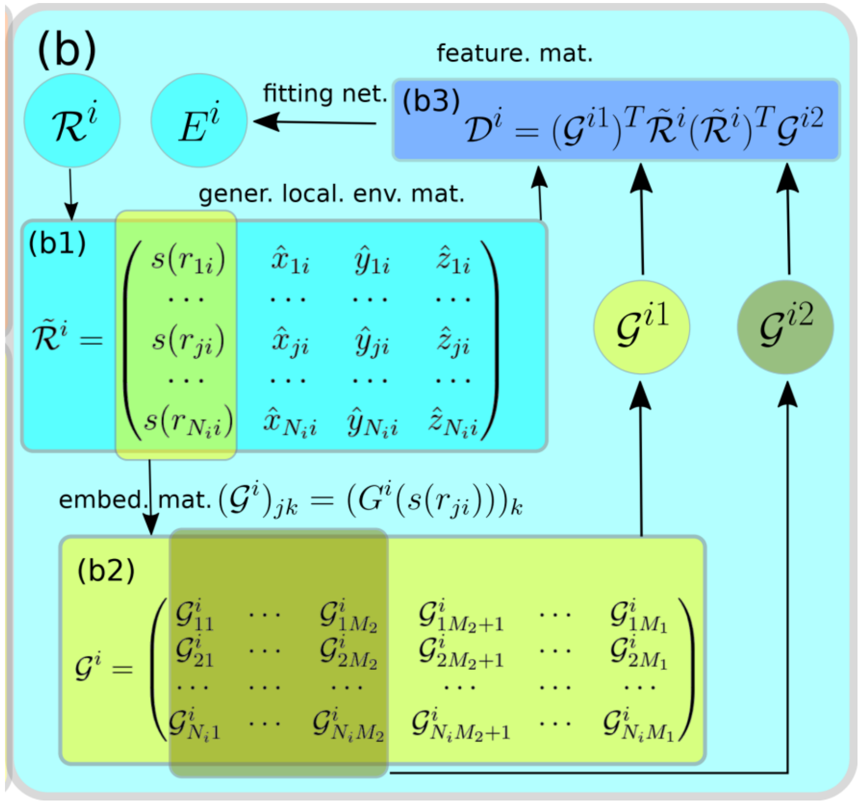
$$L(p_\epsilon, p_f, p_\xi) = \frac{p_\epsilon}{N}\Delta E^2 + \frac{p_f}{3N}\sum_i |\Delta \boldsymbol{F}_i|^2 + \frac{p_\xi}{9N}\|\Delta \Xi\|^2,$$

p_ϵ, p_f, p_ξ随训练进行，其值不断变化
$\Delta E, \Delta \boldsymbol{F}_i, \Delta\Xi$分别是三者各自的RMSE

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$$

目标：使损失函数值最小



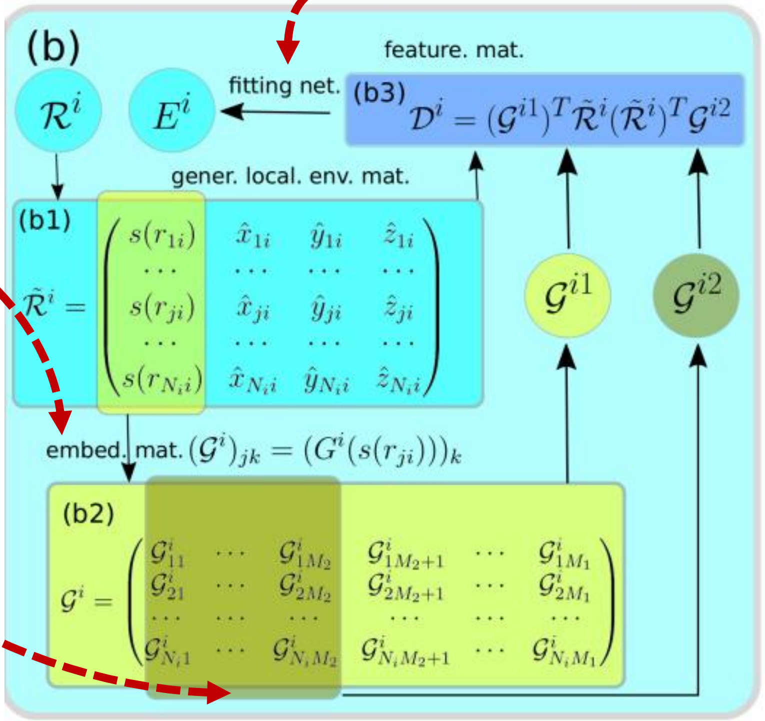

(b)
feature. mat.
fitting net.
(b3) $\mathcal{D}^i = (\mathcal{G}^{i1})^T\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T\mathcal{G}^{i2}$
$\mathcal{R}^i$
E^i
gener. local. env. mat.
(b1) $\tilde{\mathcal{R}}^i = \begin{pmatrix} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i i}) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{pmatrix}$
$\mathcal{G}^{i1}$
$\mathcal{G}^{i2}$
embed. mat. $(\mathcal{G}^i)_{jk} = (G^i(s(r_{ji})))_k$
(b2) $\mathcal{G}^i = \begin{pmatrix} \mathcal{G}^i_{11} & \cdots & \mathcal{G}^i_{1M_2} & \mathcal{G}^i_{1M_2+1} & \cdots & \mathcal{G}^i_{1M_1} \\ \mathcal{G}^i_{21} & \cdots & \mathcal{G}^i_{2M_2} & \mathcal{G}^i_{2M_2+1} & \cdots & \mathcal{G}^i_{2M_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{G}^i_{N_i 1} & \cdots & \mathcal{G}^i_{N_i M_2} & \mathcal{G}^i_{N_i M_2+1} & \cdots & \mathcal{G}^i_{N_i M_1} \end{pmatrix}$

网络结构参数

$$s(r_{ji}) = \begin{cases} \dfrac{1}{r_{ji}}, & r_{ji} < r_{cs}. \\ \dfrac{1}{r_{ji}}\left\{\dfrac{1}{2}\cos\left[\pi\dfrac{(r_{ji}-r_{cs})}{(r_c-r_{cs})}\right]+\dfrac{1}{2}\right\}, & r_{cs} < r_{ji} < r_c. \\ 0, & r_{ji} > r_c. \end{cases}$$

```
"descriptor" :{
    "type":               "se_a",
    "sel":                [46, 92],
    "rcut_smth":          5.80,
    "rcut":               6.00,
    "neuron":             [25, 50, 100],
    "resnet_dt":          false,
    "axis_neuron":        16,
    "seed":               1
```

```
"fitting_net" : {
    "neuron":            [240, 240, 240],
    "resnet_dt":         true,
    "seed":              1
}
```

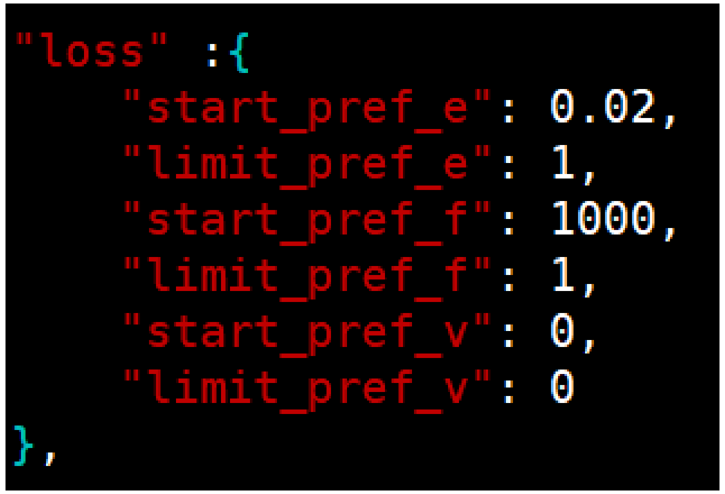


(b)
$\mathcal{R}^i$
E^i
fitting net.
feature. mat.
(b3) $\mathcal{D}^i = (\mathcal{G}^{i1})^T\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T\mathcal{G}^{i2}$
gener. local. env. mat.
(b1) $\tilde{\mathcal{R}}^i = \begin{pmatrix} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i i}) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{pmatrix}$
$\mathcal{G}^{i1}$
$\mathcal{G}^{i2}$
embed. mat. $(\mathcal{G}^i)_{jk} = (G^i(s(r_{ji})))_k$
(b2) $\mathcal{G}^i = \begin{pmatrix} \mathcal{G}^i_{11} & \cdots & \mathcal{G}^i_{1M_2} & \mathcal{G}^i_{1M_2+1} & \cdots & \mathcal{G}^i_{1M_1} \\ \mathcal{G}^i_{21} & \cdots & \mathcal{G}^i_{2M_2} & \mathcal{G}^i_{2M_2+1} & \cdots & \mathcal{G}^i_{2M_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{G}^i_{N_i1} & \cdots & \mathcal{G}^i_{N_iM_2} & \mathcal{G}^i_{N_iM_2+1} & \cdots & \mathcal{G}^i_{N_iM_1} \end{pmatrix}$

损失函数、训练参数

Loss function

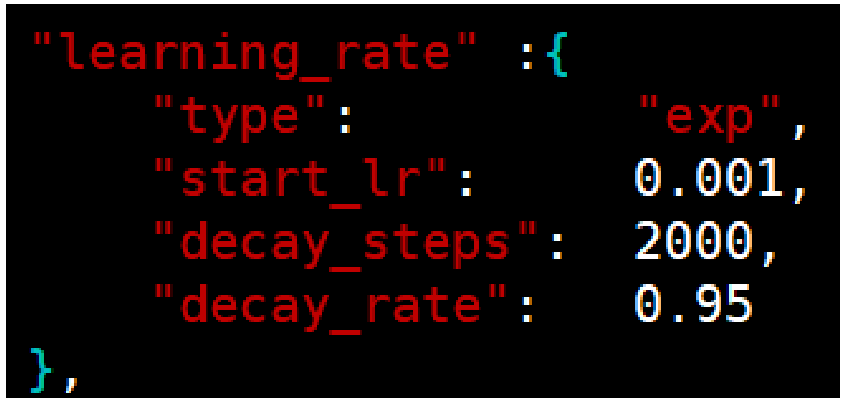
$$L(p_\epsilon, p_f, p_\xi) = \frac{p_\epsilon}{N}\Delta E^2 + \frac{p_f}{3N}\sum_i |\Delta \boldsymbol{F}_i|^2 + \frac{p_\xi}{9N}\|\Delta \Xi\|^2$$



```
"loss" :{
    "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
},
```

Learning rate

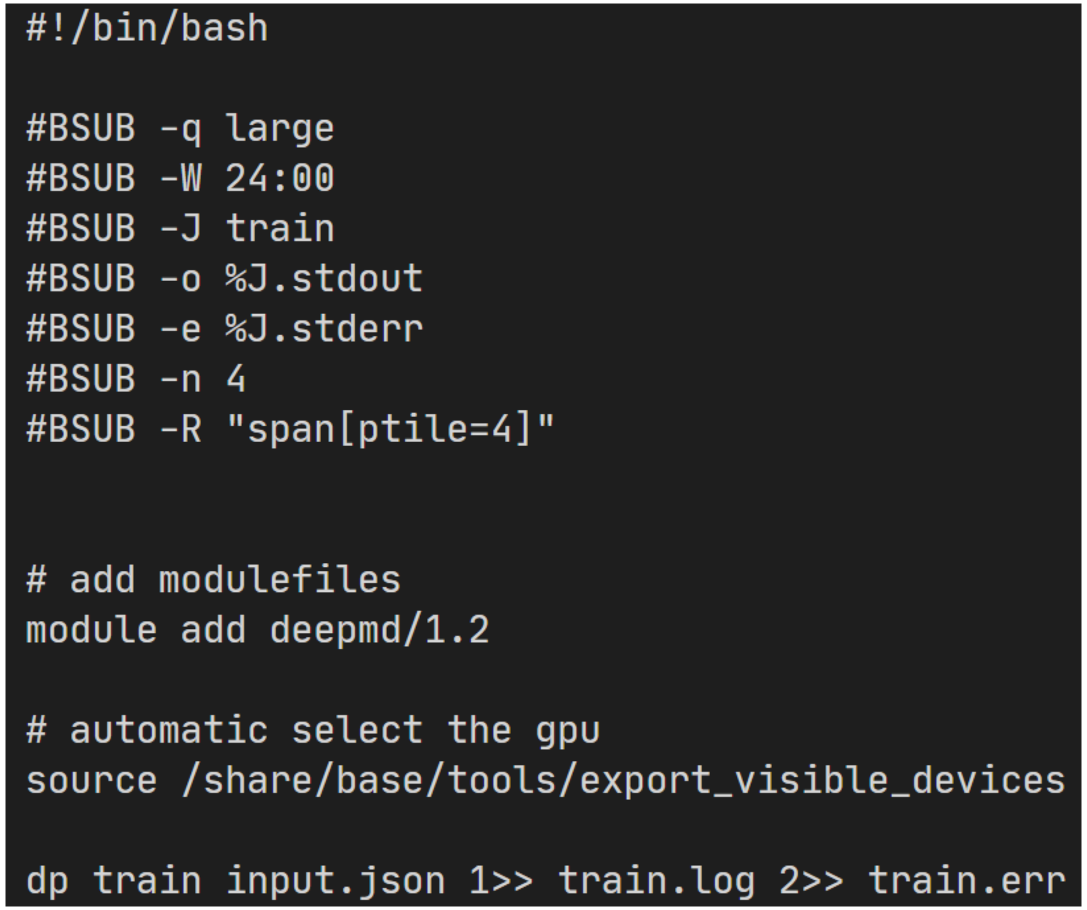
$$p(t) = p^{\text{limit}}\left[1 - \frac{r_l(t)}{r_l^0}\right] + p^{\text{start}}\left[\frac{r_l(t)}{r_l^0}\right]$$



```
"learning_rate" :{
    "type":         "exp",
    "start_lr":     0.001,
    "decay_steps":  2000,
    "decay_rate":   0.95
},
```

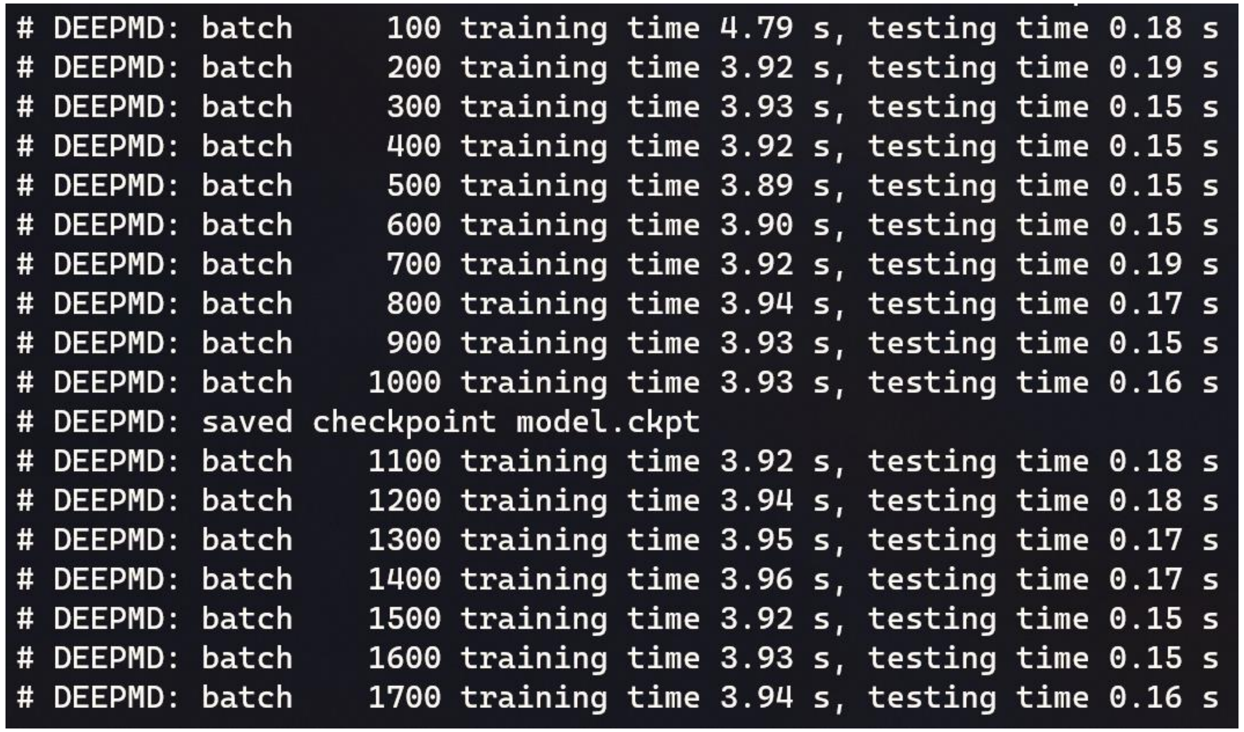
开启训练

- 作业提交脚本



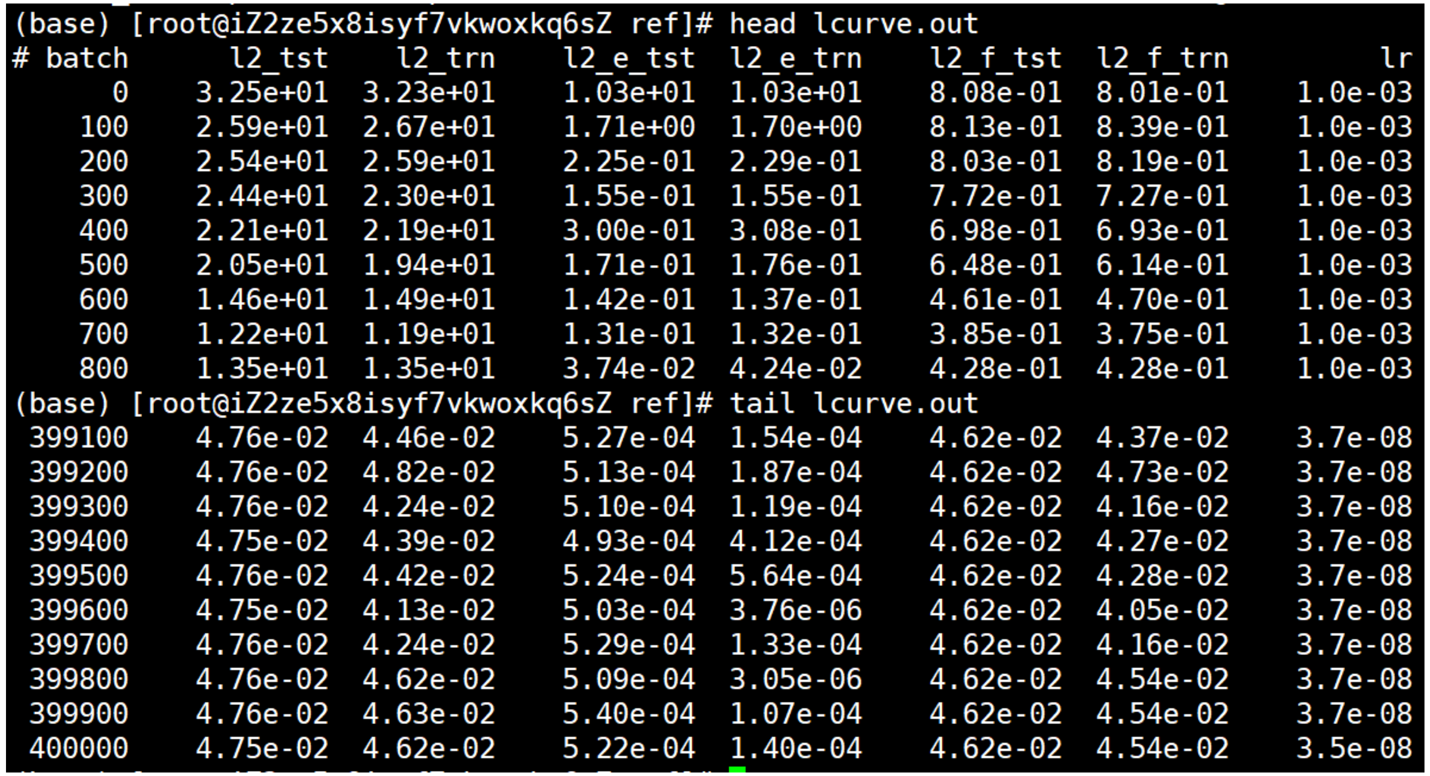
```
#!/bin/bash

#BSUB -q large
#BSUB -W 24:00
#BSUB -J train
#BSUB -o %J.stdout
#BSUB -e %J.stderr
#BSUB -n 4
#BSUB -R "span[ptile=4]"


# add modulefiles
module add deepmd/1.2

# automatic select the gpu
source /share/base/tools/export_visible_devices

dp train input.json 1>> train.log 2>> train.err
```

- 输出



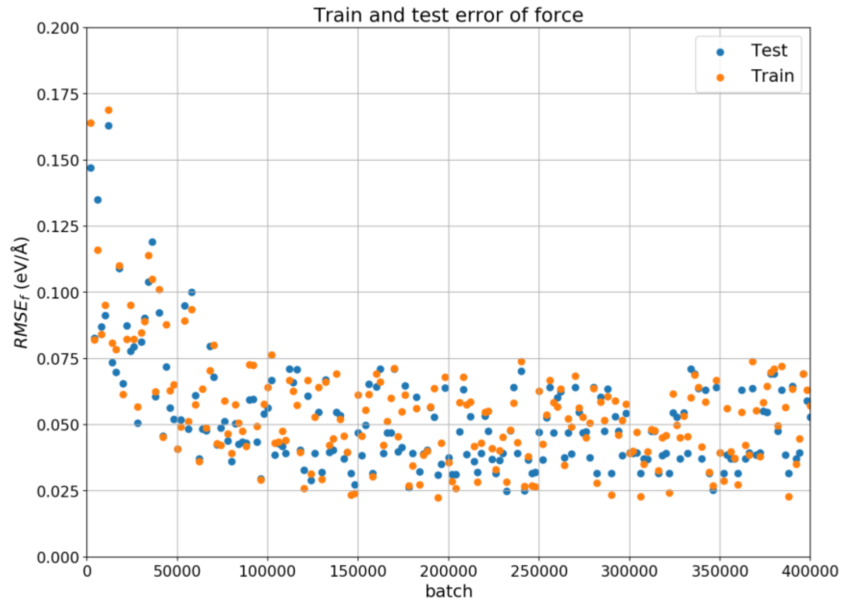
```
# DEEPMD: batch     100 training time 4.79 s, testing time 0.18 s
# DEEPMD: batch     200 training time 3.92 s, testing time 0.19 s
# DEEPMD: batch     300 training time 3.93 s, testing time 0.15 s
# DEEPMD: batch     400 training time 3.92 s, testing time 0.15 s
# DEEPMD: batch     500 training time 3.89 s, testing time 0.15 s
# DEEPMD: batch     600 training time 3.90 s, testing time 0.15 s
# DEEPMD: batch     700 training time 3.92 s, testing time 0.19 s
# DEEPMD: batch     800 training time 3.94 s, testing time 0.17 s
# DEEPMD: batch     900 training time 3.93 s, testing time 0.15 s
# DEEPMD: batch    1000 training time 3.93 s, testing time 0.16 s
# DEEPMD: saved checkpoint model.ckpt
# DEEPMD: batch    1100 training time 3.92 s, testing time 0.18 s
# DEEPMD: batch    1200 training time 3.94 s, testing time 0.18 s
# DEEPMD: batch    1300 training time 3.95 s, testing time 0.17 s
# DEEPMD: batch    1400 training time 3.96 s, testing time 0.17 s
# DEEPMD: batch    1500 training time 3.92 s, testing time 0.15 s
# DEEPMD: batch    1600 training time 3.93 s, testing time 0.15 s
# DEEPMD: batch    1700 training time 3.94 s, testing time 0.16 s
```

lcurve.out



```
(base) [root@iZ2ze5x8isyf7vkwoxkq6sZ ref]# head lcurve.out
# batch      l2_tst    l2_trn    l2_e_tst  l2_e_trn    l2_f_tst  l2_f_trn        lr
      0    3.25e+01  3.23e+01    1.03e+01  1.03e+01    8.08e-01  8.01e-01    1.0e-03
    100    2.59e+01  2.67e+01    1.71e+00  1.70e+00    8.13e-01  8.39e-01    1.0e-03
    200    2.54e+01  2.59e+01    2.25e-01  2.29e-01    8.03e-01  8.19e-01    1.0e-03
    300    2.44e+01  2.30e+01    1.55e-01  1.55e-01    7.72e-01  7.27e-01    1.0e-03
    400    2.21e+01  2.19e+01    3.00e-01  3.08e-01    6.98e-01  6.93e-01    1.0e-03
    500    2.05e+01  1.94e+01    1.71e-01  1.76e-01    6.48e-01  6.14e-01    1.0e-03
    600    1.46e+01  1.49e+01    1.42e-01  1.37e-01    4.61e-01  4.70e-01    1.0e-03
    700    1.22e+01  1.19e+01    1.31e-01  1.32e-01    3.85e-01  3.75e-01    1.0e-03
    800    1.35e+01  1.35e+01    3.74e-02  4.24e-02    4.28e-01  4.28e-01    1.0e-03
(base) [root@iZ2ze5x8isyf7vkwoxkq6sZ ref]# tail lcurve.out
 399100    4.76e-02  4.46e-02    5.27e-04  1.54e-04    4.62e-02  4.37e-02    3.7e-08
 399200    4.76e-02  4.82e-02    5.13e-04  1.87e-04    4.62e-02  4.73e-02    3.7e-08
 399300    4.76e-02  4.24e-02    5.10e-04  1.19e-04    4.62e-02  4.16e-02    3.7e-08
 399400    4.75e-02  4.39e-02    4.93e-04  4.12e-04    4.62e-02  4.27e-02    3.7e-08
 399500    4.76e-02  4.42e-02    5.24e-04  5.64e-04    4.62e-02  4.28e-02    3.7e-08
 399600    4.75e-02  4.13e-02    5.03e-04  3.76e-06    4.62e-02  4.05e-02    3.7e-08
 399700    4.76e-02  4.24e-02    5.29e-04  1.33e-04    4.62e-02  4.16e-02    3.7e-08
 399800    4.76e-02  4.62e-02    5.09e-04  3.05e-06    4.62e-02  4.54e-02    3.7e-08
 399900    4.76e-02  4.63e-02    5.40e-04  1.07e-04    4.62e-02  4.54e-02    3.7e-08
 400000    4.75e-02  4.62e-02    5.22e-04  1.40e-04    4.62e-02  4.54e-02    3.5e-08
```

$$L_2(\mathrm{E}) = \left|\left|\boldsymbol{E_{predicted}} - \boldsymbol{E_{test}}\right|\right|_2$$



l2_tst l2_trn   total error of test sets and training sets（损失函数）

l2_e_tst l2_e_trn   energy error of test sets and training sets（能量误差）

l2_f_tst l2_f_trn   force error of test sets and training sets（力误差）

固定模型

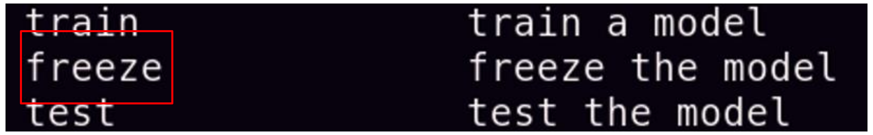
command:

```
dp freeze
```

Model file: frozen_model.pb

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

Model file: graph.pb

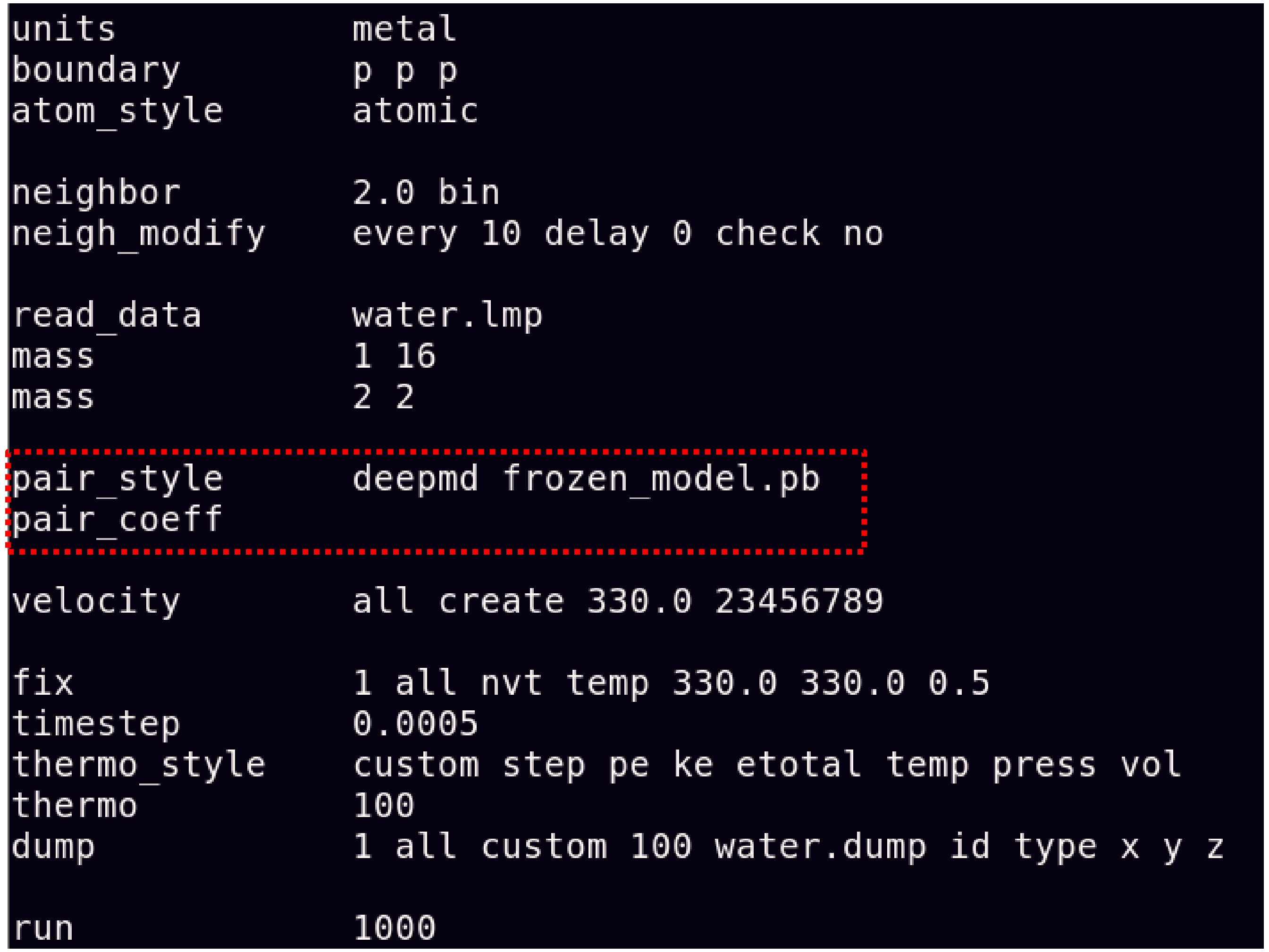


```
train                      train a model
freeze                     freeze the model
test                       test the model
```

后续在Lammps等软件中调用的模型便是这里所固定的**graph.pb**

Lammps输入文件

以水为例的输入文件示例 in.lammps



```
units           metal
boundary        p p p
atom_style      atomic

neighbor        2.0 bin
neigh_modify    every 10 delay 0 check no

read_data       water.lmp
mass            1 16
mass            2 2

pair_style      deepmd frozen_model.pb
pair_coeff

velocity        all create 330.0 23456789

fix             1 all nvt temp 330.0 330.0 0.5
timestep        0.0005
thermo_style    custom step pe ke etotal temp press vol
thermo          100
dump            1 all custom 100 water.dump id type x y z

run             1000
```

运行命令:

```
lmp_mpi -i in.lammps
```

请大家批评指正，谢谢！ | THANKS