Contents

1 MatPL

Materials Potential Library (MatPL, 原名 PWMLFF, 当前版本 MatPL-2025.3), 是一套在 GNU GPL3.0 许可下的开源软件包。

MatPL 提供了一套完备的软件、工具以及数据仓库，用于快速生成生成媲美从头算分子动力学（AIMD）的机器学习力场。包括模型训练平台 MatPL、Lammps 分子动力学接口、主动学习数据生成平台 pwact、数据格式转换工具 pwdata、数据和模型仓库。您可以通过下列链接访问它们的源码以及使用手册

1. MatPL 机器学习平台

开源仓库地址

MatPL 用于快速训练机器学习力场，这些力场的精度可以与从头算分子动力学（AIMD）相媲美

2. lammps 接口

开源仓库地址

高效的分子动力学仿真软件，无缝集成了 MatPL 的 力场模型，并支持 GPU 加速

3. 主动学习工具 pwact

开源仓库地址

PWact 是开源的基于 MatPL 的一套自动化主动学习数据生成工具。它集成了 MatPL、Lammps 接口以及常用的 PWMAT、VASP、CP2K 第一性原理软件，能够自动进行计算任务分发、监控、故障恢复、结果收集。通过使用 PWact，用户能够低成本、快速地制备覆盖广泛相空间的训练数据集

4. 结构转换工具 pwdata

开源仓库地址

pwdata 是 MatPL 的数据预处理工具，可用于提取特征和标签。同时提供 PWmat、VASP、CP2K、Lammps 间的结构格式转换以及相应的扩胞、晶格缩放、原子位置微扰操作

5. MatPL Examples

MatPL 的测试结果以及使用 MatPL 的相关案例

2 安装手册

2.1 离线和在线安装

MatPL 提供了 离线安装 和 在线安装 两种方式。对于 龙讯超算云(Mcloud) 用户、龙讯一体机用户，我们已经做了预装，只需要加载即可使用，加载方式如下所示。

2.1.1 龙讯超算云 (Mcloud) 用户

```
# 加载 MatPL
source /share/app/MATPL/MatPL-2025.3/env.sh

# 或者采用以下方式分步加载
#step1. 加载 python 运行环境
source /share/app/anaconda3/etc/profile.d/conda.sh
module load conda/3-2020.07
conda deactivate
conda activate matpl-2025.3
#step2. 加载 MatPL
module load matpl/2025.3

# 加载 lammps
module load lammps4matpl/2025.3
# 对于 Linear 和 NN 模型的 lammps 接口，我们提供了 cpu 版本的接口，使用 fortran 实现，
 ↪  请加载
module load lammps4matpl/fortran
```

2.1.2 龙讯一体机 (GPU) 用户

```
# 加载 MatPL 和 lammps
module load intel/2020 cuda/11.8
source /share/app/MATPL/MatPL-2025.3/matpl-env.sh

# 对于 Linear 和 NN 模型的 lammps 接口，我们提供了 cpu 版本的接口，使用 fortran 实现，
 ↪  请加载
module load intel/2020
source /share/app/MATPL/MatPL-2025.3/matpl-fortran-env.sh
```

2.1.3 龙讯一体机 (CPU) 用户

```
# 加载 MatPL 和 lammps
module load intel/2020
source /share/app/MATPL/MatPL-2025.3/matpl-env.sh
```

```
# 对于 Linear 和 NN 模型的 lammps 接口，我们提供了 cpu 版本的接口，使用 fortran 实现，
  ↪ 请加载
module load intel/2020
source /share/app/MATPL/MatPL-2025.3/matpl-fortran-env.sh
```

lammps 接口已经预装了下列功能

- KSPACE
- MANYBODY
- REAXFF
- MOLECULE
- QEQ
- REPLICA
- RIGID
- MEAM
- MC
- SHOCK
- MatPL

2.2 常见安装错误

2.2.1 关于编译环境

大部分的安装失败问题，都来源于编译安装环境版本不匹配，或找不到相关环境变量。请先检查下列编译器是否已正确安装，并且版本适配。

我们推荐使用 `intel2020` 版本，`cuda/11.8`，`cmake 版本 >= 3.21`，`gcc 版本 8.n`。MatPL 中使用的 `pytorch` 版本为 `2.0` 以上，必须使用 `cuda/11.8` 或更高版本。

对于 `intel/2020` 编译套件，使用了它的 `ifort` 和 `icc` 编译器 (`19.1.3`)、`mpi(2019)`、`mkl 库 (2020)`，如果单独加载，请确保版本不低于它们。

您可以通过位于源码根目录的 src/check/check_env.sh 脚本检查环境。一个正确的环境如下所示。

```
1. CUDA version is 11.8.
2. nvcc command exists.
3. ifort version is no less than 19.1, current version is 19.1.
4. MKL library is installed.
5. GCC version is not 8.x, current version is 8.
6. PyTorch is installed.
7. PyTorch version is 2.0 or above, current version is 2.2.
```

2.2.2 找不到 cuda_runtime.h 头文件

如果编译过程中找不到 cuda_runtime.h 头文件，请在 src/MAKE/Makefile.mpi 文件的 第 24 行 替换为您自己的 CUDA 路径，/the/path/cuda/cuda-11.8，cuda_runtime.h 位于该目录下的 include 目录下。

```
CUDA_HOME = $(CUDADIR)
替换为 CUDA_HOME = /the/path/cuda/cuda-11.8
```

2.2.3 NeighConst.so 编译错误

2.2.3.1 错误描述 在编译 fortran 代码过程中出现如下错误

```
ifort -O3 least_squares.f90 counts_atom.f90 scan_title.f90 transform_to_upp
      find_neighbore00.f90 find_neighbore.f90 find_feature_deepMD2.f90 \
         gen_deepMD2_feature.f90 \
    -o gen_deepMD2_feature.x -mkl
python3 -m numpy.f2py -c -m NeighConst --fcompiler=intelem --
compiler=intelem -L/share/app/intel2020ucompilers_and_libraries_2020.4.304
lmkl_rt NeighConst.f90
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/f2py/__in__.py", line 5, in <module>
    main()
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/f2py/f22e.py", line 766, in main
    run_compile()
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/f2py/f22e.py", line 594, in run_compile
    build_backend = f2py_build_generator(backend_key)
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/f2py/_bkends/__init__.py", line 6, in f2py_build_generator
    from ._distutils import DistutilsBackend
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/f2py/_bkends/_distutils.py", line 3, in <module>
    from numpy.distutils.core import setup, Extension
 File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/distuti/core.py", line 24, in <module>
```

```
    from numpy.distutils.command import config, config_compiler, \
  File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/distuti/command/config.py", line 19, in <module>
    from numpy.distutils.mingw32ccompiler import generate_manifest
  File "/data/home/wuxingxing/anaconda3/envs/pwmlff-2024.5/lib/python3.11/
packages/numpy/distuti/mingw32ccompiler.py", line 27, in <module>
    from distutils.msvccompiler import get_build_version as get_build_msvc_v
ModuleNotFoundError: No module named 'distutils.msvccompiler'
make: *** [NeighConst.so] Error 1
make: Leaving directory `/data/home/wuxingxing/codespace/PWMLFF_gpu/src/pr
make: Entering directory `/data/home/wuxingxing/codespace/PWMLFF_gpu/src/p
```

2.2.3.2 错误原因 该错误出自 setuptools 版本不匹配，一般是版本过高造成的，您需要降低 setuptools，执行如下命令：

```
# 卸载 setuptools
$ pip uninstall setuptools
# 清除本地缓存
$ pip cache purge
# 重新安装 setuptools
$ pip install setuptools==68.0.0
# 在我们的测试中不高于 68.0.0 即可
```

2.3 常见运行错误

2.3.1 环境变量检查

由于未正确加载或者未加载相关环境变量，导致的运行时错误，一般表现为找不到 MatPL 命令 或者 一些 ***.so 的动态库缺失。此时请检查下列环境变量是否都已经加载。

```
#python 环境，是否激活了 python 环境
source /the/path/etc/profile.d/conda.sh
conda activate matpl-2025.3

#intel 和 cuda 工具集是否加载
module load intel/2020 cuda/11.8

#MatPL 的环境变量是否加载
source /the/path/MatPL-2025.3/env.sh
```

2.3.2 动态库加载错误-mkl 库

```
    exec(code, run_globals)
  File "/the/path/MatPL-2025.3/main.py", line 6, in <module>
    from src.user.dp_work import dp_train, dp_test
  File "/the/path/MatPL-2025.3/src/user/dp_work.py", line 6, in
  ↪  <module>
    from src.PWMLFF.dp_network import dp_network
  File "/the/path/MatPL-2025.3/src/PWMLFF/dp_network.py", line 42, in
  ↪  <module>
    import src.pre_data.dp_mlff as dp_mlff
  File "/the/path/MatPL-2025.3/src/pre_data/dp_mlff.py", line 11, in
  ↪  <module>
    from src.lib.NeighConst import neighconst
ImportError: libmkl_rt.so: cannot open shared object file: No such
 ↪  file or directory
```

解决方法 没有加载 Intel Math Kernel Library (MKL)，intel/2020 模块 (Intel Parallel Studio XE 2020 或 Intel oneAPI Toolkits 2020 版本中的一个模块化软件) 通常包含 Intel MKL 库。加载这个模块时，MKL 库将可用于你的编译和运行环境中。

```
module load intel/2020
```

2.3.3 Lammps 接口常见运行时错误

2.3.3.1 环境变量检查 由于未正确加载或者未加载相关环境变量，导致的运行时错误，一般表现为找不到 `lmp_mpi` 命令，或者 一些 `***.so` 的动态库缺失。此时请检查下列环境变量是否都已经加载。

```
#1. 用于 mpirun 命令
module load intel/2020

#2. 加载 lammps 环境变量
source /the/path/of/lammps/env.sh

#3. 运行 lammps 命令
mpirun -np 4 lmp_mpi -in in.lammps
```

2.3.3.2 Lammps NEP 模型 nep 模型在 GPU 接口中运行一段时间后，出现如下错误：

```
    ......
    97000   1293.8659      -70999.672       35.957737      -70963.715
 ↪  13.66254        13.66254        12.50455        2334.1619
    98000   1191.7602      -71009.412       33.120127      -70976.292
 ↪  13.577541       13.577541       12.426755       2290.8676
```

```
    99000   1219.1286     -71013.893      33.880718     -70980.012
 ↪  13.488421      13.48842       12.345188      2246.0524
CUDA Error:
    File:       utilities/gpu_vector.cu
    Line:       117
    Error code: 700
    Error text: an illegal memory access was encountered

=============================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 0 PID 5490 RUNNING AT gn59
=   KILLED BY SIGNAL: 9 (Killed)
=============================================================================

=============================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 1 PID 5491 RUNNING AT gn59
=   KILLED BY SIGNAL: 9 (Killed)
=============================================================================

=============================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   RANK 2 PID 5492 RUNNING AT gn59
=   KILLED BY SIGNAL: 9 (Killed)
=============================================================================
```

解决方法 该错误一般是 MD 运行一段时间后，力场拟合精度下降，导致部分原子的邻居列表变化较大，超过了初始设置的最大邻居数目，调大最大邻居数设置即可。调整方法：在 nep 的力场文件，如下所示的 nep_to_lmps.txt：

```
nep4   2 O Si
cutoff 6.0 5.0
n_max  4 4
basis_size 12 12
l_max  4 2 1
......
```

在 `cutoff` 所在行，修改为如下

```
cutoff 6.0 5.0 500 400
```

这里 `500` 是 两体 (radial_cutoff) cutoff 对应的最大邻居数，`400` 是多体（angular_cutoff）项对应的最大邻居数。

请注意，如果出现该错误，请先检查 md 轨迹文件是否正常，是否 md 本身跑蹦了。

3 MatPL 操作命令

在 MatPL 中，您可以使用 `matpl`、`MatPL`、`MATPL` 或者 `PWMLFF` 作为起始命令。其中 `PWMLFF` 为 MatPL-2025.3 之前的版本，新版本兼容该命令。

MatPL 命令包括做训练 `train` 命令、做推理测试的 `test` 命令以及一些不同模型独有的功能性命令。您可以通过 `matpl -h` 命令输出 MatPL 的所有支持命令列表。

```
MatPL -h
或者 MatPL --help
```

3.1 train 训练

train 命令是 MatPL 的训练命令，使用该命令需要用户提前准备好训练设置的 json 文件。

```
MatPL train input.json
```

- NEP 的训练请参考 NEP 训练
- DP 的训练请参考 DP 训练
- NN 的训练请参考 NN 训练
- LINEAR 训练请参考 LINEAR 训练

train 文件目录

力场训练结束后产生如下目录如下所示

```
├── model_record
│   ├── epoch_train.dat
│   ├── epoch_valid.dat
│   ├── nep_model.ckpt
│   └── nep5.txt
├── std_input.json
├── train.json
└── forcefield/
    └── forcefield.ff
```

- `std_input.json` 为模型训练中使用的所有设置参数（用户自定义参数以及默认参数）
- `model_record/nep_model.ckpt` 为最近一个 epoch 训练结束后的力场文件，.ckpt 为 pytorch 可读的文件格式，对于 DP 力场，则为 dp_model.ckpt，对于 NN 力场则为 nn_model.ckpt
- `model_record/nep5.txt` 为 nep_model.ckpt 提取出的 txt 格式力场文件，用于 lammps 或 GPUMD 中做 MD，其他力场训练不存在该文件
- `model_record/epoch_train.dat` 为训练过程中”train_data” 中，每个 epoch 的训练集的 loss 信息汇总，内容如下所示。从左到右分别为 训练 epoch 步；总 loss；L2 loss，不开启 L2

训练则不存在该列 (ADAM 优化器对应 `lambda_2`、LKF 优化器对应 `po_weight`)；原子能量 rmse(eV/atom)；原子力 rmse(ev/Å)；原子位力 rmse(eV/atom)，不开启 `train_virial` 则不存在该列；学习率；epoch 耗时 (秒)。

```
# epoch               loss           Loss_l2   RMSE_Etot(eV/atom)
 ↪  RMSE_F(eV/Å)   RMSE_virial(eV/atom)           real_lr
 ↪  time(s)
    1    4.7907987747e+04  1.3987758802e-01     2.2508174106e+00
 ↪  7.7088011034e-01       5.4796850561e+00  1.0000000000e-03
 ↪  3.5540
    ......
```

- `model_record/epoch_valid.dat` 为训练过程中”valid_data”中，每个 epoch 训练结束时验证集的 loss 信息汇总，如果不设置验证集则不输出改文件，文件内容如下所示。从左到右分别为 训练 epoch 步；总 loss；原子能量 rmse(eV/atom)；原子力 rmse(ev/Å)；原子位力 rmse(eV/atom)，不开启 `train_virial` 则不存在该列；学习率；epoch 耗时 (秒)。

```
# epoch               loss    RMSE_Etot(eV/atom)          RMSE_F(eV/Å)
 ↪  RMSE_virial(eV/atom)
    1    2.9945036197e+04     1.8128180972e+00     6.5953191220e-01
 ↪  5.2145886493e+00
```

- `forcefield` 目录为 NN 或 linear 力场提取出的 txt 格式力场文件目录，用于fortran 版本的 lammps 接口

3.2 test 测试

test 命令是 MatPL 的测试命令，使用该命令需要用户提前准备好推理设置的 json 文件。执行成功后，该命令将输出力场对测试数据的能量和受力信息。

```
MatPL test input.json
```

- NEP 的测试请参考 NEP 测试
- DP 的测试请参考 DP 测试
- NN 的测试请参考 NN 测试
- LINEAR 测试请参考 LINEAR 测试

test 文件目录

test 结束后，在当前目录生成一个 test_result 目录，保存了 测试 结构，文件目录如下所示。

```
test_result/
│    ├──image_atom_nums.txt
│    ├── dft_total_energy.txt
│    ├── dft_force.txt
│    ├── dft_virial.txt
```

```
│   ├── dft_atomic_energy.txt
│   ├── inference_total_energy.txt
│   ├── inference_force.txt
│   ├── inference_virial.txt
│   ├── inference_atomic_energy.txt
│   ├── inference_summary.txt
│   ├── Energy.png
│   └── Force.png
└── std_input.json
```

- `image_atom_nums.txt` 存储测试集中结构对应的原子数
- `dft_total_energy.txt` 存储每个结构的能量标签
- `dft_force.txt` 存储每个结构中，每个原子的力标签，每行存储该原子的 x、y、z 三个方向分力
- `dft_virial.txt` 存储每个结构的维里标签，每个结构存储为一行，如果该结构不存在维里信息，则该行用 9 个 `-e6` 值占位
- `dft_atomic_energy.txt` 存储每个结构中，每个原子的能量标签（该标签为 PWmat 独有），每个结构存储为一行
- `inference_total_energy.txt` 存储每个结构的能量推理结果,与 dft_total_energy.txt 中的行对应
- `inference_virial.txt` 存储每个结构的维里推理结果，每个结构存储为一行，与 dft_virial.txt 中的行对应
- `inference_atomic_energy.txt` 存储每个结构中，每个原子的能量推理结果，每个结构存储为一行，与 dft_atomic_energy.txt 中的行对应
- `Energy.png` 为标签能量与力场推理能量对比
- `Force.png` 为标签受力与力场推理受力对比
- `Virial.png` 为标签维里与力场推理维里对比，标签不存在维里，则不存在该图
- `inference_summary.txt` 存储本次测试的汇总信息，如下例子中所示。

```
For 1140 images:
Average RMSE of Etot per atom: 0.029401988821789057
Average RMSE of Force: 0.045971754863441294
Average RMSE of Virial per atom: None

More details can be found under the file directory:
/the/path/test/test_result
```

3.3 其他功能性命令

MatPL 对不同的模型提供了不同的功能性命令

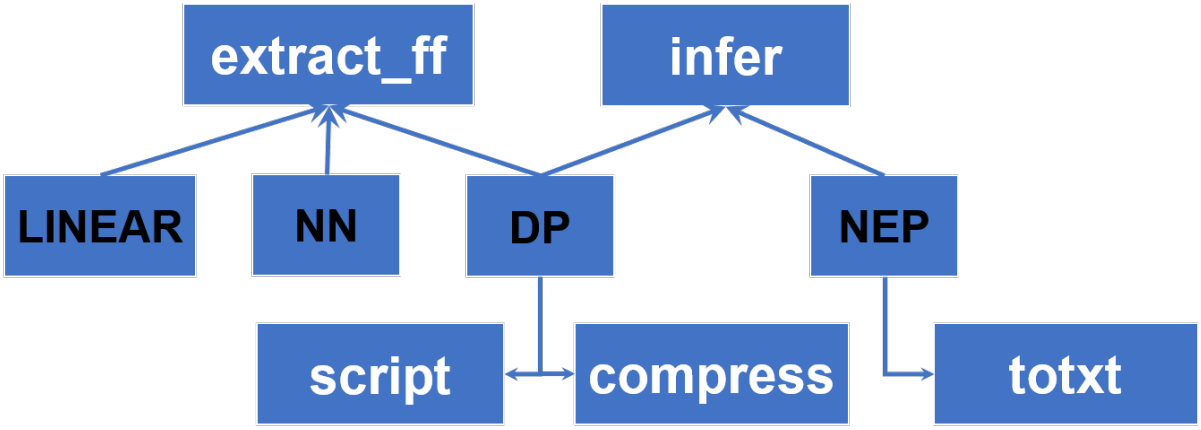


Figure 1: MatPL 功能性命令

3.3.1 extract_ff

该命令用于提取 NN 力场的 ckpt 文件为 txt 格式，提取后的力场文件可以用于 fortran 语言 实现的 lammps 接口。

```
# 提取 nn 力场模型
MatPL extract_ff nn_model.ckpt
```

操作使用请参考 - 提取 NN 力场

3.3.2 infer

该命令用于使用 NEP 或 DP 模型对单结构文件做能量和受力推理。

```
# nep 模型推理 pwmat atom.config 结构
MatPL infer nep_to_lmps.txt atom.config pwmat/config
MatPL infer nep_modek.ckpt atom.config pwmat/config
# dp 模型推理 lammps dump 结构
MatPL infer dp_model.ckpt 0.lammpstrj lammps/dump Hf O
```

操作使用请参考 - NEP 力场 单结构推理 - DP 力场 单结构推理

3.3.3 totxt

该命令为 NEP 力场独有，用于将 NEP 的 ckpt 力场文件转换为 lammps 或 GPUMD 中使用的 txt 格式力场。

```
MatPL totxt nep_model.ckpt
```

操作请参考 - NEP 力场转为 Lammps 或 GPUMD 格式

3.3.4 compress

该命令为 DP 力场独有，用于 DP 模型的推理加速，原理是将 DP 模型中的 embedding net 网络拟合为多项式，在训练集的原子类型较多时有明显的加速效果。完整的模型压缩指令如下：

```
MatPL compress dp_model.ckpt -d 0.01 -o 3 -s cmp_dp_model
```

- compress 是压缩命令
- dp_model.ckpt 为待压缩模型文件名称，为必须要提供的参数
- -d 为 S_ij 的网格划分大小，默认值为 0.01
- -o 为模型压缩阶数，3 为三阶模型压缩，5 为五阶模型压缩，默认值为 3
- -s 为压缩后的模型名称，默认名称为“cmp_dp_model”

压缩后，将在当前目录得到一个名称为 `cmp_dp_model.ckpt` 的力场文件。操作请参考 - DP 力场多项式压缩

3.3.5 script

该命令为 DP 力场独有，用于将 DP 的 ckpt 力场文件转换为 libtorch 格式，之后该文件可用于 lammps 模拟。

```
MatPL script dp_model.ckpt
# 转换后将在当前目录生成一个 jit_dp.pt 文件
# 转化压缩后的力场文件
MatPL script cmp_dp_model.ckpt
# 转换后将在当前目录生成一个 jit_cmp_dp.pt 文件
```

操作请参考 - DP 力场转 libtorch 格式

3.4 Lammps 力场应用

MatPL 提供了 lammps 力场接口，安装方式请参考 在线安装 或 离线安装

lammps 运行命令为

```
# 加载 lammps 环境变量 env.sh 文件，正确安装后，该文件位于 lammps 源码根目录下
source /the/path/of/lammps/env.sh
# 执行 lammps 命令
mpirun -np N lmp_mpi -in in.lammps
```

这里 N 为 md 中的使用的 CPU 核数，如果您的设备中存在可用的 GPU 资源（例如 M 张 GPU 卡），则在运行中，N 个 lammps 线程将平均分配到这 M 张卡上。我们建议您使用的 CPU 核数与您设置的 GPU 数量相同，多个线程在单个 GPU 上会由于资源竞争导致运行速度降低。

此外，lammps 接口允许跨节点以及跨节点 GPU 卡并行，只需要指定节点数、GPU 卡数即可。

运行 lammps 时需要在 lammps 控制文件中指定力场文件所在路径，如下所示。

```
pair_style    matpl    力场文件路径
pair_coeff    * *      8 72
```

其中：- pair_style 设置力场文件路径，这里 `matpl` 为固定格式，代表使用 MatPL 中力场

这里也支持多模型的偏差值输出，该功能一般用于主动学习采用中。您可以指定多个模型，在模拟中将使用第 1 个模型做 MD，其他模型参与偏差值计算，例如例子中所示，此时 pair_style 设置为如下:

```
pair_style    matpl    0_jit_dp.pt 1_jit_dp.pt 2_jit_dp.pt 3_jit_dp.pt
↪  out_freq DUMP_FREQ_VALUE out_file model_devi.out
```

- pair_coeff 指定待模拟结构中的原子类型对应的原子序号。例如，如果您的结构中 1 为 O 元素，2 为 Hf 元素，设置 `pair_coeff * * 8 72` 即可。

对于 DP 和 NEP 力场在 Lammps 中的设置，请参考 - NEP lammps MD - DP lammps MD

对于 NN 和 LINEAR 力场 pair_style 在 Lammps 中的设置，稍有不同：

```
pair_style    matpl
pair_coeff    * * 3 1 forcefield.ff 29
```

- pair_style 设置使用 matpl 力场
- pair_coeff 设置力场文件和原子类型。这里 3 表示使用 Neural Network 模型产生的力场，如果使用 Liear 力场，请设置为 1；第二个数字 1 表示读取 1 个力场文件，forcefield.ff 为 MatPL 生成的力场文件名称，29 为 Cu 的原子序数
- NN lammps MD
- LINEAR lammps MD

4 MatPL 可用参数

本节介绍了所有模型中可由用户定义的参数，可以分为基础参数和高级参数两类。基础参数需要用户指定，高级参数采用了默认值，用户可以在 json 文件中根据需求手动修改。在下面的参数中，“相对路径（relative path）”表示相对于当前工作目录的路径，而“绝对路径（absolute path）”表示从根目录开始的文件或目录的完整路径。

MatPL 参数可以分为基础参数和超参数两类。对于 MatPL 中的力场，只需要设置基础参数即可完成模型的训练、测试和相关分子动力学过程。超参数包括模型超参数和优化器超参数，用于对模型和训练优化器的细节设置。

4.1 基础参数

4.1.1 model_type

该参数用于指定用于训练的模型类型。您可以使用 `LINEAR` 模型、NN 模型、DP 模型或 `NEP` 模型。

4.1.2 atom_type

该参数用于设置训练体系的元素类型。用户可以按照任意顺序指定元素的原子序数。例如，对于单元素系统如铜，可以设置为 [29]，而对于多元素系统如 CH4，则可以设置为 [1, 6]。您也可以使用元素类型的名称，例如 [“Cu”] 或者 [“H”，“C”]。

4.1.3 train_data

该参数用于指定训练集数据路径。您可以使用相对路径或绝对路径。- 对于 DP 和 NEP 模型，支持的文件格式有 `extxyz`、`pwmlff/npy`、`deepmd/npy`、`deepmd/raw`、`pwmat/movement`, `vasp/outcar`, `cp2k/md` - 对于 LINEAR 和 NN 模型，仅支持 `pwmat/movement` 格式

4.1.4 valid_data

该参数用于指定验证集数据路径。您可以使用相对路径或绝对路径。- 对于 DP 和 NEP 模型，支持的文件格式有 `extxyz`、`pwmlff/npy`、`deepmd/npy`、`deepmd/raw`、`pwmat/movement`, `vasp/outcar`, `cp2k/md` - 对于 LINEAR 和 NN 模型，仅支持 `pwmat/movement` 格式

4.1.5 test_data

该参数用于 `test` 命令做推理时指定测试集数据路径。您可以使用相对路径或绝对路径。- 对于 DP 和 NEP 模型，支持的文件格式有 `extxyz`、`pwmlff/npy`、`deepmd/npy`、`deepmd/raw`、

pwmat/movement, vasp/outcar, cp2k/md - 对于 LINEAR 和 NN 模型，仅支持 pwmat/movement 格式

4.1.6 format

该参数用于指定数据 (train_data、valid_data、test_data) 的格式，支持的数据格式有扩展的 xyz 格式 extxyz 、pwmlff/npy、deepmd/npy、deepmd/raw 格式。此外也支持直接使用 PWmat, VASP, CP2K 轨迹文件，对应 format 参数分别为 pwmat/movement, vasp/outcar, cp2k/md。默认格式为 pwmat/movement。细节请参考数据格式转换工具pwdata。:::info 注意，输入数据的格式需要一致。:::

4.1.7 model_load_file

该参数用于 test 命令做推理时指定模型的路径，支持相对或者绝对路径。

4.1.8 recover_train

该参数用于从中断的训练任务中恢复训练。默认值为 true

4.1.9 reserve_work_dir

该参数用于 LINEAR 或者 NN 模型，用于指定在任务执行完成后是否保留工作目录 work_dir。默认值为 False，意味着在执行完成后该目录将被删除。### save_step 该参数用于设置每隔多少个 iteration 保存一次模型，默认值为 None，即只在每个 epoch 训练结束后保存一次模型。

4.1.10 max_save_num

该参数用与 save_step 配合使用，用于设置最多保存 max_same_num 个最近的模型。默认值为 10，仅在设置 save_step 后起作用。

4.2 NEP 模型超参数

完整的 NEP 模型参数设置如下:

```
    "model": {
        "descriptor": {
            "cutoff": [6.0,6.0],
            "n_max": [4,4],
            "basis_size": [12,12],
```

```
            "l_max": [4,2,1],
            "zbl": 2.0
        },
        "fitting_net": {
            "network_size": 40
        }
    }
```

4.2.1 cutoff

该参数用于设置 radial 和 angular 的截断能。默认值为 [8.0, 4.0]。

4.2.2 n_max

该参数用于设置 radial 和 angular 分别对应的描述符数量，该值不小于 0，不大于 19，默认值为 [4, 4]。

4.2.3 basis_size

该参数用于设置 radial 和 angular 对应的基组数量，该值不小于 0，不大于 19 默认值为 [8, 8]。

4.2.4 l_max

该参数用于设置 angular 的展开阶，同时控制是否使用四体和五体描述符，默认值为 [4, 2, 1]，分别是三体、四体以及五体 描述符 对应的阶。这里 2 表示使用四体 描述符，1 表示使用五体描述符。如果您只使用三体描述符，请设置为 [4, 0, 0]；只是用三体和四体描述符，请设置为 [4, 2, 0]。

NEP 两体描述符的数量为 n_max[0]+1；三体描述符的数量为 (n_max[1] + 1)*l_max[0]，四体描述符、五体描述符数量相同，分别为 n_max[1] + 1。

4.2.5 network_size

该参数用于设置 NEP 模型中隐藏层神经元个数，在 NEP 模型中只有一层隐藏层，默认值为 40。

4.2.6 zbl

该参数用于设置 Ziegler-Biersack-Littmark (ZBL) 势，处理原子距离非常近的情况。默认不设置。该值的允许范围是 $1.0 \leq \text{zbl} \leq 2.5$。

4.3 DP 模型超参数

DP 模型的完整参数设置如下：

```
    "type_embedding":false,
    "model": {
        "type_embedding":{
            "physical_property":["atomic_number", "atom_mass",
             ↪  "atom_radius", "molar_vol", "melting_point",
             ↪  "boiling_point", "electron_affin", "pauling"]
        },
        "descriptor": {
            "Rmax": 6.0,
            "Rmin": 0.5,
            "M2": 16,
            "network_size": [25,25,25]
        },
        "fitting_net": {
            "network_size": [50,50,50,1]
        }
    }
```

4.3.1 type_embedding

该参数用于 DP 模型训练开启 type embedding 时设置相应参数。您也可以在’ model’ 同级字典下设置`"type_embedding":true`,此时将采用 [“atomic_number”, “atom_radius”, “atom_mass” , “electron_affin” , “pauling”] 设置。默认值为 `false`，不开启 type_embdding。

4.3.1.1 physical_property 该参数用于指定 DP 模型在做 type embedding 方式训练时需要的参数，我们这里提供了 8 个物理属性供用户选择。

- atomic_number: 原子序数
- atom_mass: 原子质量
- atom_radius: 原子半径
- molar_vol: 摩尔体积
- melting_point: 熔点
- boiling_point: 沸点
- electron_affin: 电子亲和能
- pauling 为泡林电负性

"physical_property" 默认值为 ["atomic_number", "atom_radius", "atom_mass",

4.3.2 Rmax

DP 模型中平滑函数的最大截断半径。默认值为 6.0Å。

4.3.3 Rmin

DP 模型中平滑函数的最小截断半径。默认值为 0.5Å。

4.3.4 M2

该参数用于 DP 模型中的网络，确定嵌入网络的输出大小和拟合网络的输入大小。在示例中，嵌入网络的输出大小为 (25 X 16)，拟合网络的输入大小为 (25 X 16 = 400)。默认值为 16。

4.3.5 network_size

该参数用于嵌入网络（embedding_net）和拟合网络（fitting_net）的结构。默认值分别为 [25, 25, 25] 和 [50, 50, 50, 1]。对应的网络结构如下所示：

嵌入网络的结构：输入层（输入数据维度）-> 隐藏层 1（25 个神经元）-> 隐藏层 2（25 个神经元）-> 输出层 3（25 个神经元）

拟合网络的结构：输入层（M2 X 25）-> 隐藏层 1（50 个神经元）-> 隐藏层 2（50 个神经元）-> 隐藏层 3 (50 个神经元）-> 输出层（1 个神经元）

4.4 NN 模型超参数

NN 模型的完整参数设置如下：

```
    "model": {
        "descriptor": {
            "Rmax": 6.0,
            "Rmin": 0.5,
            "feature_type": [3,4]
        },
        "fitting_net": {
            "network_size": [15,15,1]
        }
    }
```

4.4.1 Rmax

特征的最大截断半径。默认值为 6.0Å。

4.4.2 Rmin

特征的最小截断半径。默认值为 0.5Å。

4.4.3 feature_type

该参数用于特征类型。支持的选项有 [1, 2]、[3, 4]、[5]、[6]、[7] 和 [8]。默认值为 [3, 4]，即 2-b 和 3-b 高斯特征。有关不同特征类型的更详细信息，请参考附录 1。

4.4.4 network_size

该参数用于拟合网络（fitting_net）的结构。默认值为 [15, 15, 1]，其结构如下所示：输入层（输入数据维度）-> 隐藏层 1（15 个神经元）-> 隐藏层 2（15 个神经元）-> 输出层（1 个神经元）

4.5 Linear 模型超参数

Linear 模型的完整参数设置如下：

```
    "model": {
        "descriptor": {
            "Rmax": 6.0,
            "Rmin": 0.5,
            "feature_type": [3,4]
        }
    }
```

4.5.1 Rmax

特征的最大截断半径。默认值为 6.0Å。

4.5.2 Rmin

特征的最小截断半径。默认值为 0.5Å。

4.5.3 feature_type

该参数用于特征类型，与 NN 模型中的设置相同。支持的选项有 [1, 2]、[3, 4]、[5]、[6]、[7] 和 [8]。默认值为 [3, 4]，即 2-b 和 3-b 高斯特征。有关不同特征类型的更详细信息，请参考附录 1。

4.6 ADAM optimizer 优化器超参数

ADAM 优化器的完整参数设置如下:

```
"optimizer": {
    "optimizer": "ADAM",
    "epochs": 30,
    "batch_size": 1,
    "print_freq": 10,
    "lambda_2" : 0.1,
    "learning_rate": 0.001,
    "stop_lr": 3.51e-08,
    "stop_step": 1000000,
    "decay_step": 5000,
    "train_energy": true,
    "train_force": true,
    "train_virial": false,
    "start_pre_fac_force": 1000,
    "start_pre_fac_etot": 0.02,
    "start_pre_fac_virial": 50.0,
    "end_pre_fac_force": 1.0,
    "end_pre_fac_etot": 1.0,
    "end_pre_fac_virial": 1.0
}
```

4.6.1 optimizer

该参数用于指定优化器名称，默认为 ADAM。对 LKF 优化器，指定名称为 'LKF'。关于优化器的详细信息参考 LKF，其中提供了有关优化器实现和特性的更深入的细节说明。

4.6.2 epochs

该参数用于指定训练的轮数（epochs）。在机器学习中，一个 epoch 指的是整个训练数据集通过神经网络的完整传递，包括前向传播和反向传播。在每个 epoch 中，训练数据集分为多个 小批量（mini-batches）样本，之后把每个批次输入到神经网络，进行前向传播、损失计算和参数更新的反向传播过程。训练的轮数决定了整个训练数据集在训练过程中被处理的次数。默认值为 30。

通常需要通过调试和评估训练过程来选择适当的训练轮数。如果训练轮数过小，模型可能无法充分学习数据集的模式和特征，导致欠拟合。另一方面，如果训练轮数过大，模型可能会过拟合训练数据，在新数据上的泛化性能下降。

4.6.3 batch_size

批大小（batch size）参数确定了在每个 epoch 的训练过程中，每个小批量（mini-batch）中包含的训练样本数量。默认值为 `1`。

4.6.4 print_freq

该参数用于指定每经过多少个小批量迭代之后打印一次训练误差。默认值为 `10`。

4.6.5 train_energy

该参数用于指定是否训练 total energy，默认值为 `true`。

4.6.6 train_force

该参数用于指定是否训练 force，默认值为 `true`。

4.6.7 train_virial

该参数用于指定是否训练 virial，默认值为 `false`。

4.6.8 lambda_2

该参数用于设置 Adam 优化器的 L2 正则化项，默认不设置。设置正则化项有助于减少模型的过拟合。

4.6.9 learning_rate

该参数是 Adam 优化器的初始学习率。默认值为 `0.001`。

4.6.10 stop_lr

该参数是指停止学习率，表示当学习率降到该值时学习率将停止更新，后续训练学习率为该值。默认值为 `3.51e-08`。

4.6.11 stop_step

该参数是指停止步数（stopping step），表示当达到该步数时学习率将停止更新，此时学习率值等于 stop_lr 指定的值。stop_step 默认值为 `1000000`。

4.6.12 decay_step

该参数表示衰减步数（decay step），它指定了学习率衰减的间隔。在每个衰减步数之后，学习率会根据一定的衰减率进行更新。默认值为 `5000`。

`learning_rate`, `stop_lr`, `stop_step`, `decay_step` 这四个变量用于更新学习率，其计算过程如下所示，可以使用以下的 Python 代码或数学公式表示：

```
decay_rate = np.exp(np.log(stop_lr/learning_rate) /
 ↪  (stop_step/decay_step))
real_lr = learning_rate * np.power(decay_rate,
 ↪  (iter_num//decay_step))
```

首先计算衰减率（decay_rate）：

$$\text{decay\_rate} = \exp\left(\frac{\log(\text{stop\_lr}/\text{start\_lr})}{\text{stop\_step}/\text{decay\_step}}\right)$$

更新学习率 learning rate：

$$\text{real\_lr} = \text{start\_lr} \cdot \text{decay\_rate}^{\lfloor \text{iter\_num}/\text{decay\_step} \rfloor}$$

其中，iter_num 代表训练过程中的迭代次数。

4.6.13 start_pre_fac_force

训练开始时 force 损失的 prefactor，应大于或等于 0。默认值为 `1000`。

4.6.14 start_pre_fac_etot

训练开始时 total energy 损失的 prefactor，应大于或等于 0。默认值为 `0.02`。

4.6.15 start_pre_fac_virial

训练开始时 virial 损失的 prefactor，应大于或等于 0。默认值为 `50.0`。

!–#### start_pre_fac_egroup 训练开始时 egroup 损失的 prefactor，应大于或等于 0。默认值为 `0.02`。

4.6.16 end_pre_fac_force

训练结束时 force 损失的 prefactor，应大于或等于 0。默认值为 `1.0`。

4.6.17 end_pre_fac_etot

训练结束时 total energy 损失的 prefactor，应大于或等于 0。默认值为 `1.0`。

4.6.18 end_pre_fac_virial

训练结束时 virial 损失的 prefactor，应大于或等于 0。默认值为 `1.0`。

4.6.19 max_norm & norm_type (按范数裁剪)

参数 max_norm 和 norm_type 配合使用，用于设置按照范数裁剪梯度。max_norm 默认值为 None，不使用按范数裁剪。

计算所有参数梯度的范数，如果超过 max_norm （max_norm 为 浮点值)，则按比例缩放梯度使范数等于 max_norm。

作用：保持梯度方向的相对关系（所有梯度同比例缩放)；适合防止梯度爆炸的同时保留梯度间的平衡；norm_type 可选（如 L2 范数、L1 范数等)。

norm_type，整形值，取值为 1 或 2，1 表示用 L1 范数，2 表示用 L2 范数。默认值为 2，启用了按范数裁剪时，将默认按 L2 范数裁剪。

L1 范数是梯度的绝对值之和：$\|g\|_1 = \sum_{i=1}^{n} |g_i|$，如果 $\|g\|_1 >$ max_norm，则梯度会被缩放为：

$$g_{\text{clipped}} = g \cdot \frac{\text{max\_norm}}{\|g\|_1}$$

L2 范数是梯度的欧几里得范数：$\|g\|_2 = \sqrt{\sum_{i=1}^{n} g_i^2}$，如果 $\|g\|_2 >$ max_norm，则梯度会被缩放为：

$$g_{\text{clipped}} = g \cdot \frac{\text{max\_norm}}{\|g\|_2}$$

4.6.20 clip_value (按值裁剪)

按值裁剪梯度。直接将所有梯度元素裁剪到 [-clip_value, clip_value] 区间，超过阈值的梯度被截断。默认值为 None, 不使用按值裁剪。

$$g_i^{(\text{clipped})} = \begin{cases} \text{clip\_value}, & \text{if } g_i > \text{clip\_value} \\ -\text{clip\_value}, & \text{if } g_i < -\text{clip\_value} \\ g_i, & \text{otherwise} \end{cases}$$

4.6.21 t_0 & t_mult

参数 t_0 和 t_mult 配合使用，用于设置在 ADAM 优化器中使用余弦退火算法更新学习率。注意：启用了余弦退火后，学习率的更新由调度器 optim.lr_scheduler.CosineAnnealingWarmRestarts 完全接管，在decay_step中的学习率更新策略将失效。

- T_0 学习率第一次回到初始值的 epoch 位置；
- T_mult 控制学习率变化的速度。如果 T_mult=1，则学习率在 T_0,2T_0,3T_0,....,i*T_0*,.... 处回到最大值 *(初始学习率)*；如果 *T_mult>1*，则学习率在 *T_0,(1+T_mult)*T_0,(1+T_mult+T_mult**2)*T_0,.....,(1+T_mult+T_mult2+...+T_0i)*T0，处回到最大值。如果开启余弦退火策略，在训练过程中，每次重启学习率前（即学习率最低点）的模型将保存在 `model_record/saved_models` 目录下。

如下图所示，该例中初始学习率 learning_rate 为 0.001，T_0 = 1, T_mult = 2, 最小学习率 stop_lr = 3.51e-08。

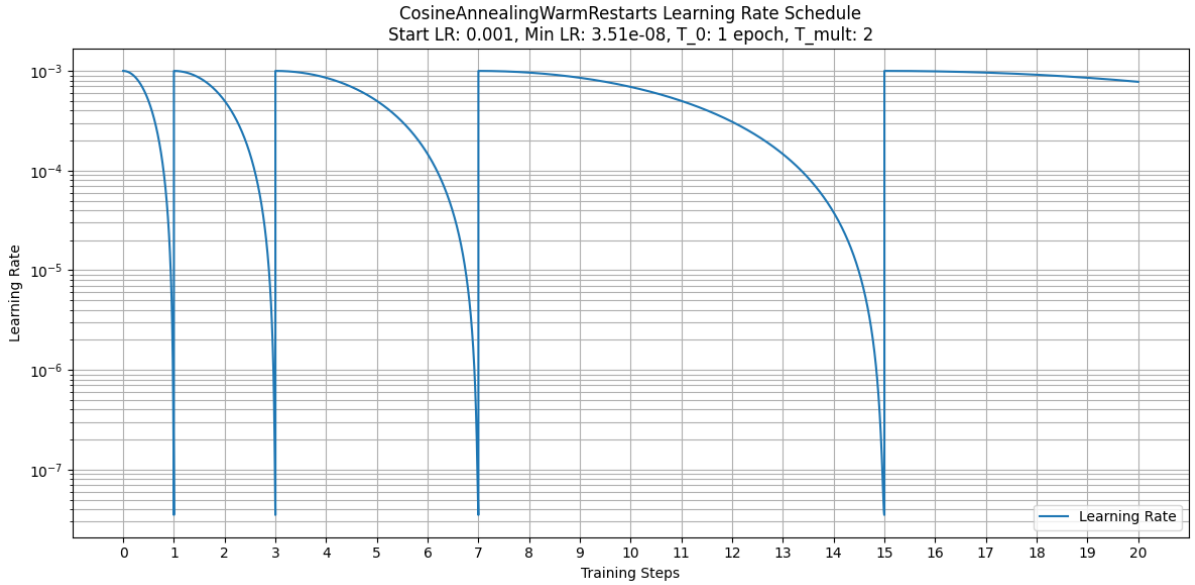


Figure 2: AL_T0_T_mult

4.7 KF optimizer 优化器超参数

KF 优化器的完整参数设置如下：

```
    "optimizer": {
        "optimizer": "LKF",
        "epochs": 30,
        "batch_size": 1,
        "print_freq": 10,
        "block_size": 5120,
```

```
        "p0_weight": 0.01,
        "kalman_lambda": 0.98,
        "kalman_nue": 0.9987,
        "train_energy": true,
        "train_force": true,
        "train_virial": false,
        "pre_fac_force": 2.0,
        "pre_fac_etot": 1.0,
        "pre_fac_virial": 1.0
    }
```

optimizer, epochs, batch_size, print_freq, train_energy, train_force, train_virial 参数与 ADAM 优化器中的参数功能相同。

4.7.1 block_size

该参数是 LKF 优化器的超参数，用于指定协方差矩阵 P 的块大小。较大的块大小会增加内存和 GPU 内存的消耗，导致训练速度较慢，而较小的块大小会影响收敛速度和准确性。默认值为 5120，如果是在 A100、H100 等高端显卡上，建议设置为 10240。

4.7.2 p0_weight

该参数是 LKF 的超参数，用于正则化参数，默认值为 0.01，即采用正则化。设置正则化项有助于减少模型的过拟合。该参数要求值小于 1，经过测试 0.01 是较为合适的值。如果设置为 1 则表示不使用正则化。

4.7.3 kalman_lambda

该参数是 LKF 的超参数，称为记忆因子（memory factor）。它决定了对先前数据的权重或关注程度。值越大，越重视先前的数据。默认值为 0.98。

4.7.4 kalman_nue

该参数是 LKF 的超参数，kalman_nue 是遗忘率（forgetting rate），描述了 kalman_lambda 变化的速率。默认值为 0.9987。

4.7.5 pre_fac_etot

该参数用于指定 total energy 对损失函数的权重或贡献。默认值为 1.0。

4.7.6 pre_fac_force

该参数用于指定 force 对损失函数的权重或贡献。默认值为 `2.0`。

4.7.7 pre_fac_virial

该参数用于指定 virial 对损失函数的权重或贡献。默认值为 `1.0`。

5 MatPL 案例实操

开源仓库地址

包括 8 种具有平移、旋转和置换不变性的特征类型

1. 2-body(2b)
2. 3-body(3b)
3. 2-body Gaussian(2b gauss)
4. 3-body Cosine(3b cos)
5. Moment Tensor Potential(MTP)
6. Spectral Neighbor Analysis Potential(SNAP)
7. DP-Chebyshev(dp1)
8. DP-Gaussian(dp2)

4 种训练模型

1. Linear
2. Neural Network(NN)
3. DP se_e2_a(Pytorch)
4. Neuroevolution Potential(NEP)

2 种高效的训练优化器

1. Adaptive Moment Estimation (ADAM)
2. Reorganized Layer Extended Kalman Filtering (LKF)

5.1 NEP 模型

操作演示

5.1.1 模型介绍

NEP 模型最初是在 GPUMD 软件包中实现的 (2022b GPUMD)。GPUMD 中训练 NEP 采用了可分离自然演化策略 (separable natural evolution strategy，SNES)，由于不依赖梯度信息，实现简单。但是对于标准的监督学习任务，特别是深度学习，更适合采用基于梯度的优化算法。我们在 `MatPL 2025.3` 版本实现了 NEP 模型（NEP5，网络结构如图 1 所示)，能够使用 MatPL 中基于梯度的 `LKF` 或 `ADAM` 优化器做模型训练。并且我们对梯度计算中的耗时部分通过 c++ cuda 算子做了优化，大幅提升了训练速度。

我们在多种体系中比较了 LKF 和 SNES 两种优化方法的训练效率，测试结果表明，LKF 优化器在对 NEP 模型的训练中展现了优越的训练精度和收敛速度 NEP 模型的网络结构只有一个单隐藏层，具有非常快的推理速度，而引入 LKF 优化器则大幅提高了训练效率。用户可以在 MatPL 中以较低的训练代价获得优质的 NEP 并使用它进行高效的机器学习分子动力学模拟，这对于资源/预算有限的用户非常友好。

我们也实现了 NEP 模型的 Lammps 分子动力学接口，支持 CPU 或 GPU 设备，受益于 NEP 简单的网络结构和化繁为简的 feature 设计，NEP 模型在 lammps 推理中具有非常快的速度，并支持跨节点（跨节点 GPU)。

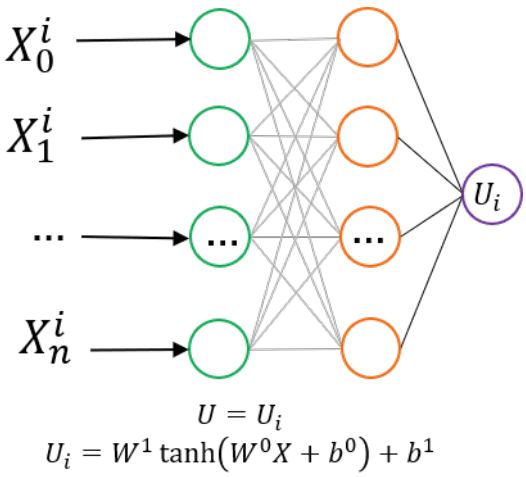


Figure 3: NEP 网络结构

NEP 网络结构，不同类型的元素具有独立但结构相同的子神经网络。此外，与文献中 NEP4 网络结构不同的是，对于每层的 bias，所有的 子网络不共享最后一层 `bias`，在 GPUMD 中为 NEP5。

5.1.2 lammps 接口测试

下图展示了 NEP 模型的 lammps CPU 和 GPU 接口在 `3090*4` 机器上做 NPT 系综 MD 模拟的速度。对于 CPU 接口，速度正比与原子规模和 CPU 核数；对于 GPU 接口, 速度正比与原子规模和 GPU 数量。

根据测试结果，我们建议如果您需要模拟的体系规模在 10^3 量级以下，建议您使用 CPU 接口即可。另外使用 GPU 接口时，建议您使用的 CPU 核数与 GPU 卡数相同。

atom nums HfO2 NPT	lammps NEP CPU version timesteps/s			lammps NEP GPU version timesteps/s	
	1 CPU	4CPU	28CPU	1GPU	4GPU
96	46.707	-	1362.395	427.945	661.099
1.2w	-	2.194	13.754	116.389	247.883
9.6w	0.0691	-	1.858	13.069	51.62
32.4w	-	-	0.566	3.715	13.485
259.2w	-	-	0.07	0.425	1.548
614.4w	-	-	0.0293	显存耗尽	0.636
1200w	-	-	-	-	0.247

Figure 4: NEP lammps 速度测试

5.2 NEP 操作演示

这里，我们以 MatPL 源码根目录/example/HfO2/nep_demo 为例 (HfO2 训练集来源)，演示 NEP 模型的训练、测试、lammps 模拟以及其他功能。案例目录结构如下所示。

```
HfO2/
├── atom.config
├── pwdata/
└── nep_demo/
    ├── nep_test.json
    ├── nep_train.json
    ├── train.job
    └── nep_lmps/
        ├── in.lammps
        ├── lmp.config
        ├── nep_to_lmps.txt
        ├── runcpu.job
        └── rungpu.job
```

- pwdata 目录为训练数据目录
- nep_train.json 是训练 NEP 力场输入参数文件
- nep_test.json 是测试 NEP 力场输入参数文件
- train.job 是 slurm 提交训练任务例子
- nep_lmps 目录下 为 NEP 力场的 lammps md 例子
  - 力场文件 nep_to_lmps.txt
  - 初始结构 lmp.config

- 控制文件 in.lammps
- runcpu.job 和 rungpu.job 是 slurm 脚本例子

5.2.1 train 训练

在 nep_demo 目录下使用如下命令即可开始训练：

```
MatPL train nep_train.json
# 或修改环境变量之后通过 slurm 提交训练任务 sbatch train.job
```

输入文件解释

nep_train.json 中的内容如下所示，关于 NEP 的参数解释，请参考 NEP 参数手册：

```
{
    "model_type": "NEP",
    "atom_type": [
        8, 72
    ],
    "optimizer": {
        "optimizer": "ADAM",
        "epochs": 30,
        "batch_size": 1,
        "print_freq": 10,
        "train_energy": true,
        "train_force": true,
        "train_virial": true
    },

    "format": "pwmlff/npy",
    "train_data": [
        "../pwdata/init_000_50/", "../pwdata/init_002_50/",
        "../pwdata/init_004_50/", "../pwdata/init_006_50/",
        "../pwdata/init_008_50/", "../pwdata/init_010_50/",
        "../pwdata/init_012_50/", "../pwdata/init_014_50/",
        "../pwdata/init_016_50/", "../pwdata/init_018_50/",
        "../pwdata/init_020_20/", "../pwdata/init_022_20/",
        "../pwdata/init_024_20/", "../pwdata/init_026_20/",
        "../pwdata/init_001_50/", "../pwdata/init_003_50/",
        "../pwdata/init_005_50/", "../pwdata/init_007_50/",
        "../pwdata/init_009_50/", "../pwdata/init_011_50/",
        "../pwdata/init_013_50/", "../pwdata/init_015_30/",
        "../pwdata/init_017_50/", "../pwdata/init_019_50/",
        "../pwdata/init_021_20/", "../pwdata/init_023_20/",
        "../pwdata/init_025_20/", "../pwdata/init_027_20/"
```

```
    ],
    "valid_data":[
        "../pwdata/init_000_50/", "../pwdata/init_004_50/",
        "../pwdata/init_008_50/"
    ]
}
```

训练结束后的力场文件目录请参考 model_record 详解

5.2.2 test 测试

test 命令支持来自 MatPL `nep_model.ckpt` 力场文件，以及在 lammps 或 GPUMD 中使用的 `nep5.txt` 格式文件。

```
MatPL test nep_test.json
```

test.json 中的内容如下所示，参数解释请参考 参数手册

```
{
    "model_type": "NEP",
    "format": "pwmlff/npy",
    "model_load_file": "./model_record/nep_model.ckpt",
    "test_data": [
        "../init_000_50", "../init_004_50", "../init_008_50",
        "../init_012_50", "../init_016_50", "../init_020_20",
        "../init_024_20", "../init_001_50", "../init_005_50",
        "../init_009_50", "../init_013_50", "../init_017_50",
        "../init_021_20", "../init_025_20", "../init_002_50",
        "../init_006_50", "../init_010_50", "../init_014_50",
        "../init_018_50", "../init_022_20", "../init_026_20",
        "../init_003_50", "../init_007_50", "../init_011_50",
        "../init_015_30", "../init_019_50", "../init_023_20",
        "../init_027_20"
    ]
}
```

测试结束后的力场文件目录请参考 test_result 详解

5.2.3 infer 推理单结构

infer 命令支持来自 MatPL `nep_model.ckpt` 力场文件、GPUMD 的 `nep4.txt` 文件、lammps 和 GPUMD 中通用的 `nep5.txt` 格式文件。

```
MatPL infer nep_model.ckpt atom.config pwmat/config
MatPL infer gpumd_nep.txt 0.lammpstrj lammps/dump Hf O
```

```
# Hf O 为 lammps/dump 格式的结构中的元素名称，Hf 为结构中 1 号元素类型，O 为元素中 2 号元
↪ 素类型
```

推理成功后，将在窗口输出推理的总能、每原子能量、每原子受力和维里

5.2.4 totxt 转 ckpt 训练文件为 nep5.txt

用于把 MatPL 训练的 nep_model.ckpt 文件转换为 txt 格式的 nep5.txt 文件，该文件可用于 GPUMD 或 lammps-MatPL 中做分子动力学模拟。

```
MatPL totxt nep_model.ckpt
```

执行成功将在执行该命令的所在目录生成名称为 nep5.txt 文件

5.2.5 lammps MD

step1. 准备力场文件

将训练完成后生成的 nep_model.ckpt 力场文件用于 lammps 模拟，您需要 提取力场文件，您只需要输入如下命令

```
MatPL totxt nep_model.ckpt
```

转换成功之后，您将得到一个力场文件 nep5.txt。

如果您的模型正常训练结束，在 model_record 目录下会存在一个 nep5.txt 文件，您可以直接使用。

此外，也支持 GPUMD 的 NEP5、 NEP4 力场文件。

step2. 准备输入控制文件

您需要在 lammps 的输入控制文件中设置如下力场,这里以 HfO2 为例(Hf02/nep_demo/nep_lmps

```
pair_style    matpl    nep_to_lmps.txt
pair_coeff    * *      8 72
```

- pair_style 设置力场文件路径，这里 matpl 为固定格式，代表使用 MatPL 中力场，nep_to_lmps.txt 为力场文件路径

  这里也支持多模型的偏差值输出，该功能一般用于主动学习采用中。您可以指定多个模型，在模拟中将使用第 1 个模型做 MD，其他模型参与偏差值计算，例如例子中所示，此时 pair_style 设置为如下:

  ```
  pair_style    matpl    0_nep_to_lmps.txt 1_nep_to_lmps.txt
  ↪  2_nep_to_lmps.txt 3_nep_to_lmps.txt  out_freq ${DUMP_FREQ}
  ↪  out_file model_devi.out
  pair_coeff    * *      8 72
  ```

- pair_coeff 指定待模拟结构中的原子类型对应的原子序号。例如，如果您的结构中 1 为 O 元素，2 为 Hf 元素，设置 `pair_coeff * * 8 72` 即可。

这里也可以将 `nep_to_lmps.txt` 文件替换为您的 GPUMD 中的 NEP4 或 NEP5 力场文件。

step3 启动 lammps 模拟

```
# 加载 lammps 环境变量 env.sh 文件，正确安装后，该文件位于 lammps 源码根目录下
source /the/path/of/lammps/env.sh
# 执行 lammps 命令
mpirun -np N lmp_mpi -in in.lammps
```

这里 N 为 md 中的使用的 CPU 核数，如果您的设备中存在可用的 GPU 资源（例如 M 张 GPU 卡），则在运行中，N 个 lammps 线程将平均分配到这 M 张卡上。我们建议您使用的 CPU 核数与您设置的 GPU 数量相同，多个线程在单个 GPU 上会由于资源竞争导致运行速度降低。

此外，lammps 接口允许跨节点以及跨节点 GPU 卡并行，只需要指定节点数、GPU 卡数即可。

5.2.6 ASE 接口

NEP 模型提供了 ase 接口，使用方式如下脚本例子所示gitee 或 github。

```
from src.ase.calculate import MatPL_calculator
calc = MatPL(model_file='nep_model.ckpt or nep.txt')
atoms = ..... # create ase.atoms.Atoms
atoms.calc = calc # or atoms.set_calculator(calc)
energy = atoms.get_potential_energy()
forces = atoms.get_forces()
stress = atoms.get_stress()
```

注意，在使用本 ase 接口时确保已经导入了MatPL 的环境变量。

5.3 DP 模型

操作演示

5.3.1 DP 模型介绍

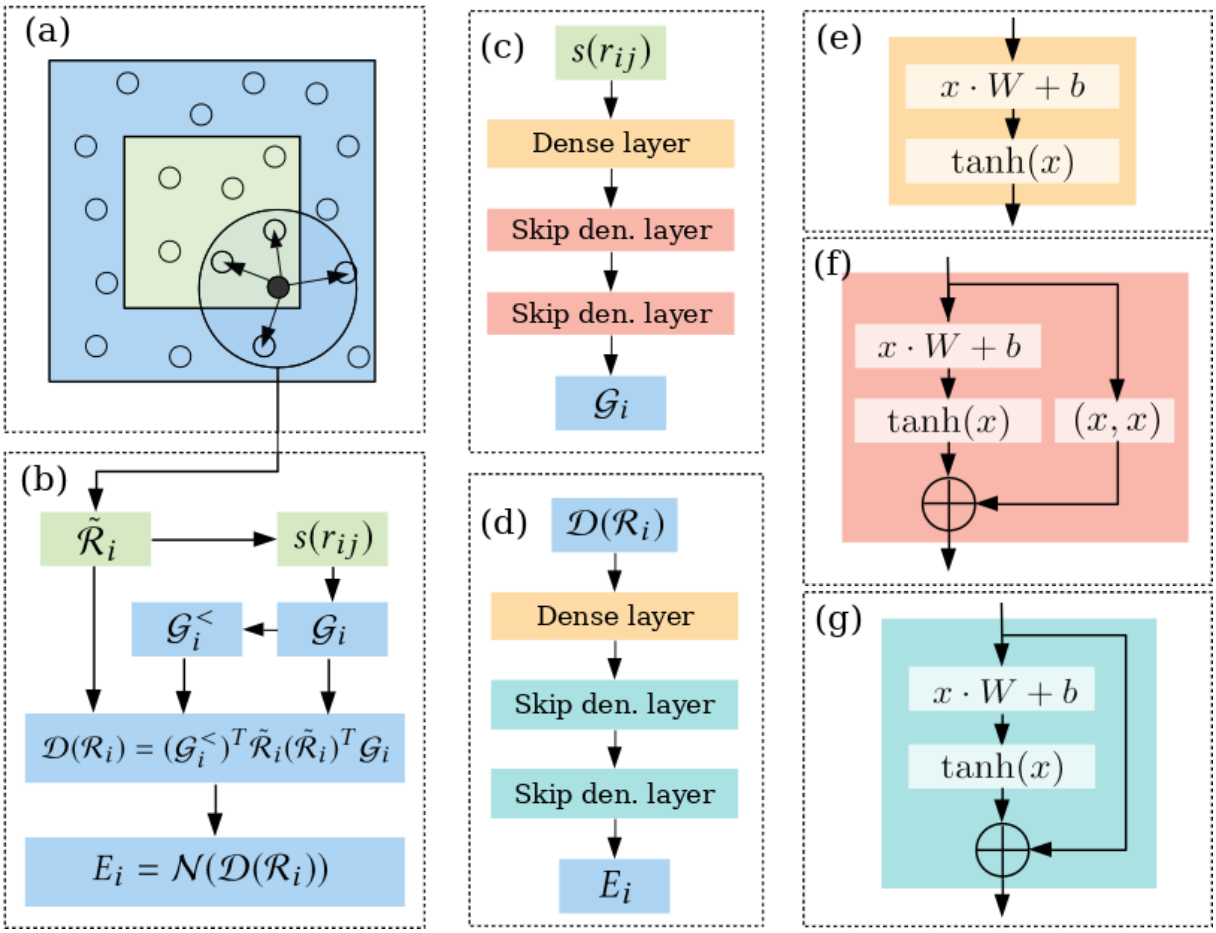


Figure 5: DP 网络结构

DP 模型请参考文献:

- [SC′ 20] Weile Jia, Han Wang, Mohan Chen, Denghui Lu, Lin Lin, Roberto Car, E Weinan, Linfeng Zhang*, "Pushing the Limit of Molecular Dynamics with Ab Initio Accuracy to 100 Million Atoms with Machine Learning," SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1-14, doi: 10.1109/SC41405.2020.00009.(CCF-A)(Gordon Bell Prize)
- Han Wang, Linfeng Zhang, Jiequn Han, and Weinan E. "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics." Computer Physics Communications 228 (2018): 178-184. doi:10.1016/j.cpc.2018.03.016
- Zhang L, Han J, Wang H, et al. End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems[J]. Advances in neural information processing systems, 2018, 31.
- Lu D, Jiang W, Chen Y, et al. DP compress: A model compression scheme for generating efficient deep potential models[J]. Journal of chemical theory and computation, 2022,

18(9): 5559-5567.

5.3.2 type embedding

由于 DP 模型的 Embedding Net 数目是元素类型数目 N 的 N^2 倍。一方面，当体系中元素类型较多时制约了模型的训练速度，以及推理速度。另一方面，这也制约了 DP 模型在通用大模型方面的潜力。考虑到 N^2 个 Embedding net 其实隐含了对元素类型的编码，因此我们通过调整 S_{ij}，将元素类型的物理属性信息与 S_{ij} 做拼接，则只需要一个 Embedding net 即可达到与 N^2 相似效果。

对于 S_{ij}，i 为中心原子，这里将 j 对应的元素类型的物理属性与 S_{ij} 做拼接，组成一个长度为 1+ 物理属性数量的 Vector 送入 Embedding Net。在我们五元合金 (钌、铑、铱、钯、镍) 数据集以及LiGePS 四元数据集 (1200K)的测试中，基于这种 Type embedding 方法的 DP 模型，能够在达到或者超过标准的 DP 模型预测精度的同时，对训练时间减少 27%，详细结果见性能测试。

使用方法

用户只需要在控制训练的 json 文件中加入 $type\_embedding$ 参数，即可开启模型训练，将使用默认物理属性训练，参见项目案例 **example/LiGePS/ligeps.json**。

```
{
  "type_embedding": true
}
```

用户也可以在该 Json 文件的 model 参数 中指定所需要的物理属性。

5.3.3 性能测试-精度

五元合金混合数据集 (9486 个构型)下，Type embedding 方法相对于标准的 DP 模型在验证集上的预测精度对比:

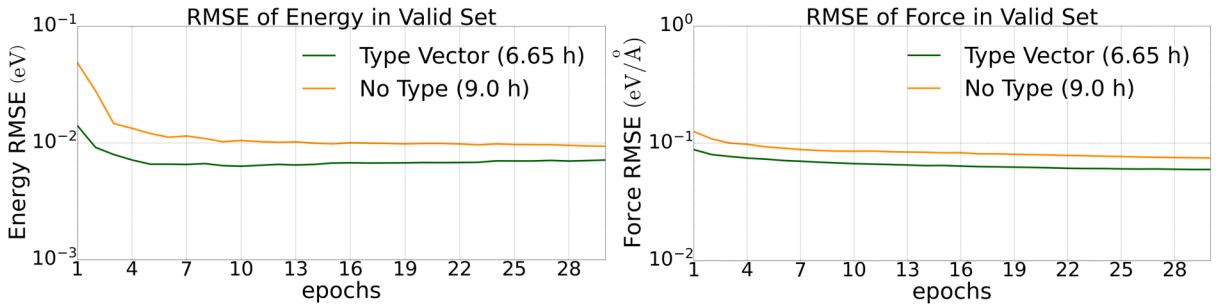


Figure 6: 五元合金体系验证集上的能量和受力误差下降

四元 LiGePS 构型的数据集（10000 个构型 1200K）下 Type embedding 方法相对于标准的 DP 模型在验证集上的预测精度对比:

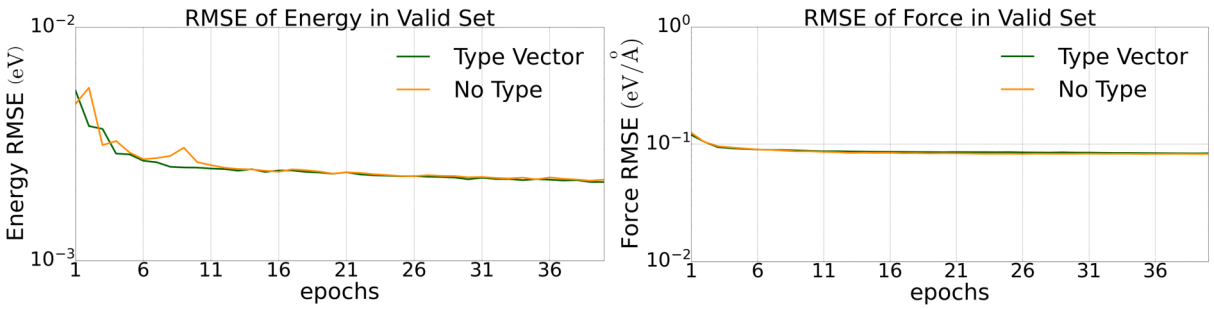


Figure 7: 四元 LiGePS 体系验证集上的能量和受力误差下降

5.3.4 性能测试-训练时间

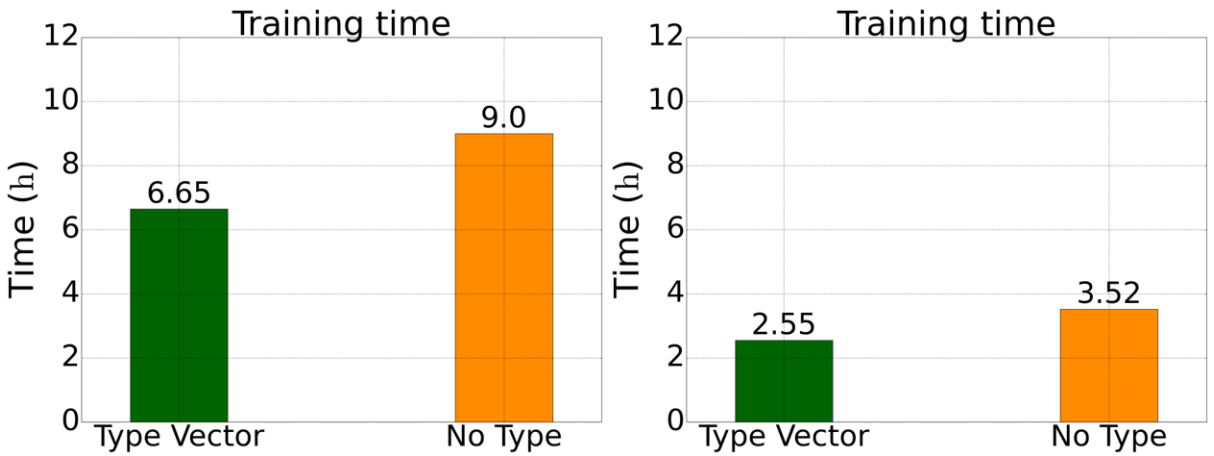


Figure 8: 四元 LiGePS 体系验证集上的能量和受力误差下降

在 Lammps 中的力场调用方式与前述标准的 DP 模型调用方法相同。

5.3.5 多项式模型压缩

DP 模型的 Embedding net 网络数目是原子类型数目 N 的 N^2 倍，随着原子类型增多，Embedding net 数目会快速增加，导致用于反向传播求导的计算图的规模会增加，成为 DP 模型做推理的瓶颈之一。如下我们对于一个五元合金系统在 DP 模型的推理过程的时间统计所示，对于 Embedding net 计算以及梯度计算的时间占比超过 90%，这存在大量的优化空间。Embedding net 的输入为一个 S_{ij} 的单值，输出为 m 个值（m 为 Embedding net 最后一层神经元数目）。因此，可以将 Embedding net 通过 m 个单值函数代替。

这里实现论文Lu D, Jiang W, Chen Y, et al. DP compress: A model compression scheme for generating efficient deep potential models中使用的五阶多项式压缩方法，同时我们也提供了基于 Hermite 插值方法的三阶多项式压缩方法供用户自由选择。

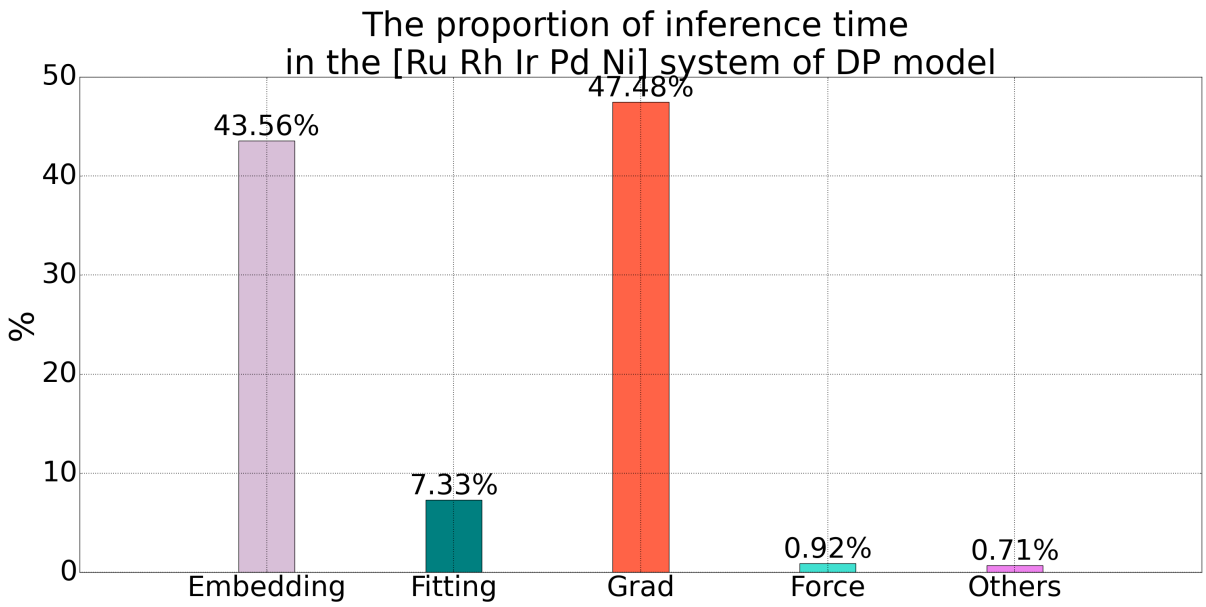


Figure 9: proportion_time

使用方法

对于一个训练后 DP 模型做模型压缩，完整的模型压缩指令如下：

```
MatPL compress dp_model.ckpt -d 0.01 -o 3 -s cmp_dp_model
```

- compress 是压缩命令
- dp_model.ckpt 为待压缩模型文件名称，为必须要提供的参数
- -d 为 S_ij 的网格划分大小，默认值为 0.01
- -o 为模型压缩阶数，3 为三阶模型压缩，5 为五阶模型压缩，默认值为 3
- -s 为压缩后的模型名称，默认名称为“cmp_dp_model”

模型压缩之后，在 lammps 中做分子动力学模拟使用方式与标准的DP 模型相同。

模型压缩精度

我们在 Bulk 铜和五元合金体系上对 DP 模型做了模型压缩，并在测试集上分别做了测试。结果如下图中所示，对于铜体系，我们加入了对二阶插值方法的精度对比，相比于三阶和五阶方法，二阶方法的精度达不到要求。

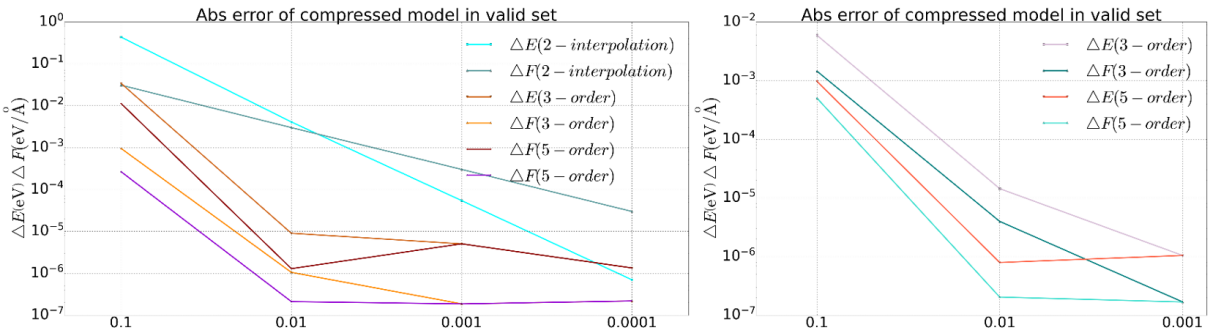


图1. Bulk铜体系DP模型二阶、三阶与五阶多项式压缩对比;图2. 五元合金体系DP模型三阶与五阶多项式压缩对比

Figure 10: 模型压缩不同阶精度对比

5.3.6 多项式模型压缩过程

5.3.6.1 网格划分 我们扫描全部训练集，得到 s_{ij} 的最大值，由于 s_{ij} 是原子 i 和 j 的三维坐标距离 r_{ij} 函数，当 $r_{ij} = r_{\text{cut}}$ 时取最小值。根据 s_{ij} 取值范围按照 dx 值等分为 L 份，则共有 $L+1$ 个插值点，分别记为 $x_1, x_2, \cdots, x_{L+1}$。在实际的使用中，由于训练集的不完备，可能存在一些 s_{ij} 值超出训练集之外，这里我们在上述网格之外，继续增加了 s_{ij} 到 $10 \times s_{ij}$ 的网格，网格大小设置为 $10 \times dx$。

5.3.6.2 三阶多项式 对于每个 $[x_l, x_{l+1})$ 区间，采用如下的三阶多项式替代 Embedding net:

$$g_m^l(x) = a_m^l x^3 + b_m^l x^2 + c_m^l x + d_m^l$$

这里 m 为 Embedding net 最后一层神经元数量，即 Embedding net 输出值数目，多项式的自变量 x 值应为 $s_{ij} - x_l$。在每个网格点上，都需要满足如下两个限定条件：

1. 多项式值与 Embedding net 输出值一致：

$$y_l = \mathcal{G}_m(x_l)$$

2. 多项式一阶导数与 Embedding net 对 s_{ij} 的一阶导一致：

$$y_l' = \mathcal{G}_m'(x_l)$$

解得对应系数为

$$a_m^l = \frac{1}{\Delta t^3}\left[(y_{l+1}' + y_l')\Delta t - 2h\right]$$

$$b_m^l = \frac{1}{\Delta t^2}\left[-(y_{l+1}' + 2y_l')\Delta t + 3h\right]$$

$$c_m^l = y_l'$$

$$d_m^l = y_l$$

其中 $h = y_{l+1} - y_l$，$\Delta t = x_{l+1} - x_l$。

5.3.6.3 五阶多项式

我们也实现了 DP Compress 中的五阶多项式压缩方法。

对于五阶多项式，对 s_{ij} 的划分方法与三阶方法相同，采用如下的多项式代替 Embedding net：

$$g_m^l(x) = a_m^l x^5 + b_m^l x^4 + c_m^l x^3 + d_m^l x^2 + e_m^l x + f_m^l$$

注意：此时多项式的自变量 x 值应为 $s_{ij} - x_l$。在每个网格点上，都需要满足如下三个限定条件：

1. 多项式值与 Embedding net 输出值一致：

$$y_l = \mathcal{G}_m(x_l)$$

2. 多项式一阶导数与 Embedding net 对 s_{ij} 的一阶导一致：

$$y_l' = \mathcal{G}_m'(x_l)$$

3. 多项式二阶导数与 Embedding net 对 s_{ij} 的二阶导一致：

$$y_l'' = \mathcal{G}_m''(x_l)$$

由此可得六个系数值分别为：

$$a_m^l = \frac{1}{2\Delta t^5}\left[12h - 6(y_{l+1}' + y_l')\Delta t + (y_{l+1}'' - y_l'')\Delta t^2\right]$$

$$b_m^l = \frac{1}{2\Delta t^4}\left[-30h + (14y_{l+1}' + 16y_l')\Delta t + (-2y_{l+1}'' + 3y_l'')\Delta t^2\right]$$

$$c_m^l = \frac{1}{2\Delta t^3}\left[20h - (8y_{l+1}' + 12y_l')\Delta t + (y_{l+1}'' - 3y_l'')\Delta t^2\right]$$

$$d_m^l = \frac{1}{2}y_l''$$

$$e_m^l = y_l'$$

$$f_m^l = y_l$$

其中 $h = y_{l+1} - y_l$，$\Delta t = x_{l+1} - x_l$。

5.3.6.4 模型压缩公式验证 Model compress 方案，将 s_{ij} 取值范围分成 L 等份，则共有 $L+1$ 个插值点，分别记为 $x_1, x_2, \cdots, x_{L+1}$。对于每个 $[x_l, x_{l+1})$ 区间，采用如下的五阶多项式替代 embedding network：

$$g_m^l(x) = a_m^l x^5 + b_m^l x^4 + c_m^l x^3 + d_m^l x^2 + e_m^l x + f_m^l$$

注意：此时多项式的自变量 x 值应为 $s_{ij} - x_l$。在每个网格点上，都需要满足如下三个边界条件：

1. 函数值一致：

$$y_l = \mathcal{G}_m(x_l)$$

2. 函数一阶导数一致：

$$y_l' = \mathcal{G}_m'(x_l)$$

3. 函数二阶导数一致：

$$y_l'' = \mathcal{G}_m''(x_l)$$

由此可得六个系数值分别为：

$$a_m^l = \frac{1}{2\Delta t^5}\left[12h - 6(y_{l+1}' + y_l')\Delta t + (y_{l+1}'' - y_l'')\Delta t^2\right]$$

$$b_m^l = \frac{1}{2\Delta t^4}\left[-30h + (14y_{l+1}' + 16y_l')\Delta t + (-2y_{l+1}'' + 3y_l'')\Delta t^2\right]$$

$$c_m^l = \frac{1}{2\Delta t^3}\left[20h - (8y_{l+1}' + 12y_l')\Delta t + (y_{l+1}'' - 3y_l'')\Delta t^2\right]$$

$$d_m^l = \frac{1}{2}y_l''$$

$$e_m^l = y_l'$$

$$f_m^l = y_l$$

其中 $h = y_{l+1} - y_l$，$\Delta t = x_{l+1} - x_l$。

5.4 DP 操作演示

这里，我们以 MatPL [源码根目录/example/HfO2/dp_demo] 为例（HfO2 训练集来源），演示 DP 模型的训练、测试、lammps 模拟以及其他功能。案例目录结构如下所示。

```
HfO2/
├── atom.config
├── pwdata/
└── dp_demo/
    ├── dp_test.json
    ├── dp_train.json
    ├── train.job
    └── dp_lmps/
        ├── in.lammps
        ├── lmp.config
        ├── jit_dp.pt
        ├── runcpu.job
        └── rungpu.job
```

- pwdata 目录为训练数据目录
- dp_train.json 是训练 DP 力场输入参数文件
- dp_train.json 是测试 DP 力场输入参数文件
- train.job 是 slurm 提交训练任务例子
- dp_lmps 目录下为 DP 力场的 lammps md 例子
  - 力场文件 jit_dp.pt
  - 初始结构 lmp.config
  - 控制文件 in.lammps
  - runcpu.job 和 rungpu.job 是 slurm 脚本例子

5.4.1 train 训练

在 dp_demo 目录下使用如下命令即可开始训练：

```
MatPL train dp_train.json
# 或修改环境变量之后通过 slurm 提交训练任务 sbatch train.job
```

输入文件解释

dp_train.json 中的内容如下所示，关于 DP 的参数解释，请参考 DP 参数手册：

```
{
    "model_type": "DP",
    "atom_type": [
        8, 72
```

```
    ],
    "format": "pwmlff/npy",
    "train_data": [
        "../pwdata/init_000_50/", "../pwdata/init_002_50/",
        "../pwdata/init_004_50/", "../pwdata/init_006_50/",
        "../pwdata/init_008_50/", "../pwdata/init_010_50/",
        "../pwdata/init_012_50/", "../pwdata/init_014_50/",
        "../pwdata/init_016_50/", "../pwdata/init_018_50/",
        "../pwdata/init_020_20/", "../pwdata/init_022_20/",
        "../pwdata/init_024_20/", "../pwdata/init_026_20/",
        "../pwdata/init_001_50/", "../pwdata/init_003_50/",
        "../pwdata/init_005_50/", "../pwdata/init_007_50/",
        "../pwdata/init_009_50/", "../pwdata/init_011_50/",
        "../pwdata/init_013_50/", "../pwdata/init_015_30/",
        "../pwdata/init_017_50/", "../pwdata/init_019_50/",
        "../pwdata/init_021_20/", "../pwdata/init_023_20/",
        "../pwdata/init_025_20/", "../pwdata/init_027_20/"
    ],
    "valid_data":[
        "../pwdata/init_000_50/", "../pwdata/init_004_50/",
        "../pwdata/init_008_50/"
    ]
}
```

训练结束后的力场文件目录请参考 model_record 详解

5.4.2 test 测试

```
MatPL test dp_test.json
```

test.json 中的内容如下所示，参数解释请参考 参数手册

```
{
    "model_type": "DP",
    "format": "pwmlff/npy",
    "model_load_file": "./model_record/dp_model.ckpt",
    "test_data": [
        "../init_000_50", "../init_004_50", "../init_008_50",
        "../init_012_50", "../init_016_50", "../init_020_20",
        "../init_024_20", "../init_001_50", "../init_005_50",
        "../init_009_50", "../init_013_50", "../init_017_50",
        "../init_021_20", "../init_025_20", "../init_002_50",
        "../init_006_50", "../init_010_50", "../init_014_50",
        "../init_018_50", "../init_022_20", "../init_026_20",
```

```
        "../init_003_50", "../init_007_50", "../init_011_50",
        "../init_015_30", "../init_019_50", "../init_023_20",
        "../init_027_20"
    ]
}
```

测试结束后的力场文件目录请参考 test_result 详解

5.4.3 infer 推理单结构

```
MatPL infer dp_model.ckpt atom.config pwmat/config
MatPL infer dp_model.ckpt 0.lammpstrj lammps/dump Hf O
# Hf O 为 lammps/dump 格式的结构中的元素名称，Hf 为结构中 1 号元素类型，O 为元素中 2 号元
  ↪ 素类型
```

推理成功后，将在窗口输出推理的总能、每原子能量、每原子受力和维里

5.4.4 compress 模型压缩

对于一个训练后 DP 力场做模型压缩，完整的模型压缩指令如下：

```
MatPL compress dp_model.ckpt -d 0.01 -o 3 -s cmp_dp_model
```

- compress 是压缩命令
- dp_model.ckpt 为待压缩模型文件名称，为必须要提供的参数
- -d 为 S_ij 的网格划分大小，默认值为 0.01
- -o 为模型压缩阶数，3 为三阶模型压缩，5 为五阶模型压缩，默认值为 3
- -s 为压缩后的模型名称，默认名称为“cmp_dp_model”

压缩后，将在当前目录得到一个名称为 `cmp_dp_model.ckpt` 的力场文件。

5.4.5 script 转 MD 力场

本命令用于将 dp_model.ckpt 文件转换为 lammps 中可识别的 libtorch 格式。

```
MatPL script dp_model.ckpt
#  或转换经过模型压缩后的力场
MatPL script cmp_dp_model.ckpt
```

转换后将在当前目录下生成一个 `jit_dp.pt` 文件，改文件可用于后续的 lammps md。

5.4.6 lammps MD

step1. 准备力场文件

将训练完成后生成的 `dp_model.ckpt` 力场文件用于 lammps 模拟，您需要 提取力场文件，您只需要输入如下命令

```
MatPL script dp_model.ckpt
```

转换成功之后，将得到一个力场文件 `jit_dp.pt`。

step2. 准备输入控制文件

您需要在 lammps 的输入控制文件中设置如下力场,这里以 HfO2 为例(`Hf02/dp_demo/dp_lmps`

```
pair_style   matpl   jit_dp.pt
pair_coeff   * *     8 72
```

其中: - pair_style 设置力场文件路径，这里 `matpl` 为固定格式，代表使用 MatPL 中力场，`jit_dp.pt` 为力场文件路径

这里也支持多模型的偏差值输出，该功能一般用于主动学习采用中。您可以指定多个模型，在模拟中将使用第 1 个模型做 MD，其他模型参与偏差值计算，例如例子中所示，此时 pair_style 设置为如下:
`txt pair_style matpl 0_jit_dp.pt 1_jit_dp.pt 2_jit_dp.pt 3_jit_dp.pt out_freq ${DUMP_FREQ} out_file model_devi.out pair_coeff * * 8 72`

- pair_coeff 指定待模拟结构中的原子类型对应的原子序号。例如，如果您的结构中 1 为 O 元素，2 为 Hf 元素，设置 `pair_coeff * * 8 72` 即可。

step3 启动 lammps 模拟

```
# 加载 lammps 环境变量 env.sh 文件，正确安装后，该文件位于 lammps 源码根目录下
source /the/path/of/lammps/env.sh
# 执行 lammps 命令
mpirun -np N lmp_mpi -in in.lammps
```

这里 N 为 md 中的使用的 CPU 核数，如果您的设备中存在可用的 GPU 资源（例如 M 张 GPU 卡），则在运行中，N 个 lammps 线程将平均分配到这 M 张卡上。我们建议您使用的 CPU 核数与您设置的 GPU 数量相同，多个线程在单个 GPU 上会由于资源竞争导致运行速度降低。

此外，lammps 接口允许跨节点以及跨节点 GPU 卡并行，只需要指定节点数、GPU 卡数即可。

5.4.7 ASE 接口

DP 模型提供了 ase 接口，使用方式如下脚本例子所示gitee 或 github。

```
from src.ase.calculate import MatPL_calculator
calc = MatPL(model_file='dp_model.ckpt')
atoms = ..... # create ase.atoms.Atoms
atoms.calc = calc # or atoms.set_calculator(calc)
energy = atoms.get_potential_energy()
forces = atoms.get_forces()
stress = atoms.get_stress()
```

注意，在使用本 ase 接口时确保已经导入了MatPL 的环境变量。

5.5 NN 模型

操作演示

5.5.1 Neural Network(NN) 模型介绍

Neural Network(NN) 中实现了如下 8 种具有平移、旋转和置换不变性的特征类型

1. 2-body(2b)
2. 3-body(3b)
3. 2-body Gaussian(2b gauss)
4. 3-body Cosine(3b cos)
5. Moment Tensor Potential(MTP)
6. Spectral Neighbor Analysis Potential(SNAP)
7. DP-Chebyshev(dp1)
8. DP-Gaussian(dp2)

特征（或描述符）是描述原子局部环境的量。它们需要保持平移、旋转和置换对称性。特征通常用作各种回归器（线性模型、神经网络等）的输入，这些回归器输出原子能量和力。由于特征是空间坐标的可微函数，因此可以计算力：

$$\mathbf{F}_i = -\frac{dE_{\text{tot}}}{d\mathbf{R}_i} = -\sum_{j,\alpha} \frac{\partial E_j}{\partial G_{j,\alpha}} \frac{\partial G_{j,\alpha}}{\partial \mathbf{R}_i}$$

其中，j 是在截断半径内的近邻原子的索引，α 是特征的索引。

5.5.2 2-b and 3-b features with piecewise cosine functions (feature 1 & 2)

给定一个中心原子，利用分段余弦函数来描述其局部环境。通过下面的图表，可以大致了解它们的原理。

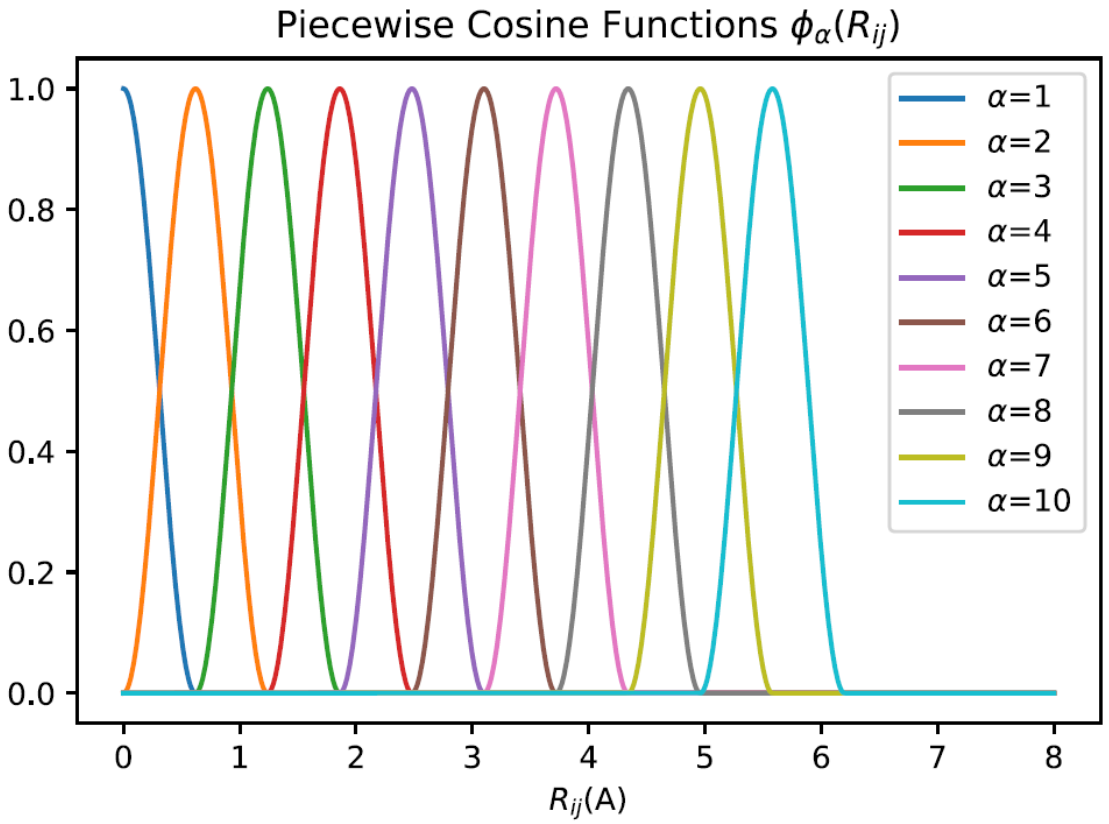


Figure 11: features

我们首先定义分段余弦函数，分别用于两体和三体特征。给定内部和外部截断 R_{inner} 和 R_{outer}，基函数的阶数 M，分段函数的宽度 h，以及中心原子 i 和近邻原子 j 之间的原子间距 R_{ij}，我们定义基函数为

$$\phi_\alpha(R_{ij}) = \begin{cases} \frac{1}{2}\cos\left(\frac{R_{ij}-R_\alpha}{h}\pi\right)+\frac{1}{2} & ,|R_{ij}-R_\alpha|<h \\ 0 & ,\text{otherwise} \end{cases}$$

其中

$$R_\alpha = R_{\text{inner}} + (\alpha-1)h,\ \alpha = 1,2,...,M$$

中心原子 i 的 **两体特征** 表达式为

$$G_{\alpha,i} = \sum_m \phi_\alpha(R_{ij})$$

而 **三体特征** 表达式为

$$G_{\alpha\beta\gamma,i} = \sum_{j,k} \phi_\alpha(R_{ij})\phi_\beta(R_{ik})\phi_\gamma(R_{jk})$$

其中 $\sum_m$ 和 $\sum_{m,n}$ 分别表示在中心原子 i 的截断半径 R_{outer} 内的所有原子的求和。

这两个特征通常是成对使用的。

参考文献：

Huang, Y., Kang, J., Goddard, W. A. & Wang, L.-W. Density functional theory based neural network force fields from energy decompositions. Phys. Rev. B 99, 064103 (2019)

5.5.3 2-b and 3-b Gaussian feature (feature 3 & 4)

这两个特征是 Behler-Parrinello 神经网络中首次使用的特征。给定截断半径 R_c 及中心原子 i 和近邻原子 j 之间的原子间距 R_{ij}，定义截断函数 f_c

$$f_c(R_{ij}) = \begin{cases} \frac{1}{2}\cos\left(\frac{\pi R_{ij}}{R_c}\right) + \frac{1}{2} & , R_{ij} < R_c \\ 0 & , \text{otherwise} \end{cases}$$

中心原子 i 的 **两体高斯** 特征定义为

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

其中 η 和 R_s 是用户定义的参数。

中心原子 i 的 **三体高斯** 特征定义为

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

其中

$$\cos\theta_{ijk} = \frac{\mathbf{R}_{ij} \cdot \mathbf{R}_{ik}}{|\mathbf{R}_{ij}||\mathbf{R}_{ik}|}$$

η、ζ 和 $\lambda = \pm 1$ 是用户定义的参数。

这两个特征通常是成对使用的。

参考文献：

J. Behler and M. Parrinello, Generalized Neural-Network Representation of High Dimensional Potential-Energy Surfaces. Phys. Rev. Lett. 98, 146401 (2007)

5.5.4 Moment Tensor Potential (feature 5)

在 MTP 中，中心原子 i 的局部环境由

$$\mathbf{n}_i = (z_i, z_j, \mathbf{r}_{ij})$$

定义，其中 z_i 是中心原子的原子类型，z_j 是近邻原子的原子类型，$\mathbf{r}_{ij}$ 是近邻原子的相对坐标。接下来，每个原子的能量贡献被展开为

$$E_i(\mathbf{n}_i) = \sum_\alpha c_\alpha B_\alpha(\mathbf{n}_i)$$

其中 B_α 是用户选择的基函数，c_α 是待拟合的参数。

为了构造基函数，我们引入矩张量 $M_{\mu\nu}$ 来定义基函数

$$M_{\mu\nu}(\mathbf{n}_i) = \sum_j f_\mu(|\mathbf{r}_{ij}|, z_i, z_j) \bigotimes_\nu \mathbf{r}_{ij}$$

这些矩张量包含径向和角度部分。径向部分可以展开为

$$f_\mu(|\mathbf{r}_{ij}|, z_i, z_j) = \sum_\beta c^{(\beta)}_{\mu, z_i, z_j} Q^{(\beta)}(|\mathbf{r}_{ij}|)$$

其中 $Q^{(\beta)}(|\mathbf{r}_{ij}|)$ 是径向基函数。具体地，

$$Q^{(\beta)}(|\mathbf{r}_{ij}|) = \begin{cases} \phi^{(\beta)}(|\mathbf{r}_{ij}|)(R_{\text{cut}} - |\mathbf{r}_{ij}|)^2 & , |\mathbf{r}_{ij}| < R_{\text{cut}} \\ 0 & , \text{otherwise} \end{cases}$$

其中 $\phi^{(\beta)}$ 是定义在区间 $[R_{\text{min}}, R_{\text{cut}}]$ 上的多项式（例如切比雪夫多项式）。

角度部分由 $\bigotimes_\nu \mathbf{r}_{ij}$ 给出，它表示对 $\mathbf{r}_{ij}$ 进行 ν 次张量积，包含了近邻 $\mathbf{n}_i$ 的角度信息。ν 决定了矩张量的秩。当 $\nu = 0$ 时，得到一个常数标量；当 $\nu = 1$ 时，得到一个向量（秩-1 张量）；当 $\nu = 2$ 时，得到一个矩阵（秩-2 张量）；以此类推。

最后，我们定义矩张量的级数为

$$\text{lev}(M_{\mu\nu}) = 2 + 4\mu + \nu$$

这是一个经验公式。

参考文献：

I.S. Novikov, etal, The MLIP package: moment tensor potential with MPI and active learning. Mach. Learn.: Sci. Technol, 2, 025002 (2021)

5.5.5 Spectral Neighbor Analysis Potential (feature 6)

在 SNAP 中，不使用高斯基函数。因此不计算两个原子局域环境图之间的距离和核函数。它首先定义一个原子局域环境，然后使用球谐函数（或 4D 球，带有旋转矩阵）来展开原子局域环境。然后使用双谱，使其具有旋转不变性。从某种意义上说，它类似于 MTP，但它使用一种特殊的方法来缩并方向指数，使其具有旋转不变性。它通常与线性回归一起使用。

首先，它定义位于 $\mathbf{r}$ 处的中心原子 i 的邻近原子周围的原子局域环境为三维空间中的 δ 函数之和：

$$\rho(\mathbf{r}) = \delta(\mathbf{r}) + \sum_{\mathbf{r}_{ki}<R_C} f_C(\mathbf{r}_{ki})\omega_k\delta(\mathbf{r}-\mathbf{r}_{ki})$$

其中 $\mathbf{r}_{ki}$ 是原子 i 的第 k 个近邻的位置，ω_k 是第 k 个近邻的权重，径向函数 $f_C(\mathbf{r}_{ki})$ 确保每个近邻的贡献在截断半径 R_C 附近平滑地变为零：

$$f_C(\mathbf{r}) = 0.5\left[\cos\left(\frac{\pi r}{R_C}\right)+1\right]$$

这个原子局域环境函数的角部分可以用球谐函数展开，球谐函数定义在 $l=0,1,2,...$ 和 $m=-l,-l+1,...,l-1,l$ 的基础上。径向分布通常由一组径向基函数表示。然而，在这里，径向信息 $\mathbf{r}$ 被映射到 4D 超球面函数 $U^j_{mm'}(\theta_0,\theta,\phi)$ 中，其中所有点（邻近原子）落入 3D 球面（在 4D 空间中），定向（角度）信息由三个角度给出：

$$\mathbf{r}\equiv\begin{pmatrix}x\\y\\z\end{pmatrix}\begin{matrix}\\\rightarrow\\\\\end{matrix}\begin{matrix}\phi=\arctan(y/x)\\\theta=\arccos(z/r)\\\theta_0=\frac{3}{4}\pi r/r_c\end{matrix}$$

因此，上述原子局域环境函数可以用这些 4D 超球面函数 $U^j_{mm'}(\theta_0,\theta,\phi)$ 展开，展开系数为 $u^j_{mm'}$：

$$\rho(\mathbf{r}) = \sum_{j=0,\frac{1}{2},1,...}^{\infty}\sum_{m=-j,-j+1}^{j}\sum_{m'=-j,-j+1}^{j} u^j_{mm'}U^j_{mm'}(\theta_0,\theta,\phi)$$

使用上述原子局域环境函数，可以计算 $u^j_{mm'}$：

$$u^j_{mm'} = U^j_{mm'}(0,0,0) + \sum_{\mathbf{r}_{ki}<R_C} f_C(\mathbf{r}_{ki})\omega_k U^j_{mm'}(\theta_0(k),\theta(k),\phi(k))$$

其中，k 是邻近原子的索引，$\theta_0(k),\theta(k),\phi(k)$ 是原子 i 的第 k 个近邻的位置矢量的三个角度。注意，$u^j_{mm'}$ 是由于其指数 m,m' 而具有方向依赖性。它们可以与下面的缩并公式（使用三个 $u^j_{mm'}$）缩并：

$$F(j_1,j_2,j)=\sum_{m_1,m_1'=-j_1}^{j}\sum_{m_2,m_2'=-j_2}^{j}\sum_{m,m'=-j}^{j}(u^{j}_{mm'})^{*}u^{j_1}_{m_1m_1'}u^{j_2}_{m_2m_2'}\times C^{jm}_{j_1m_1j_2m_2}C^{jm'}_{j_1m_1'j_2m_2'}$$

这里，$C^{jm}_{j_1m_1j_2m_2}C^{jm}_{j_1m_1'j_2m_2'}$ 是 Clebsch-Gordan 系数，最终的标量特征是 $F(j_1,j_2,j)$。通过设置不同的 j_1,j_2,j，可以产生不同的特征。注意，在这些特征中，没有径向函数索引，而是有三个角动量索引。这是因为我们已经将径向距离信息转换为 3D 球面中的第三个角度信息。

5.5.6 DP-Chebyshev (feature 7)

这个特征类似于 DP 的嵌入网络。它使用切比雪夫多项式作为基础。

首先，我们将 $S(\mathbf{r}_{ij})$ 定义为加权的距离的倒数函数：

$$S(\mathbf{r})=\frac{f_C(\mathbf{r})}{r}$$

$$f_C(\mathbf{r})=\begin{cases}1 & ,r<R_{C_2}\\ \frac{1}{2}\cos\left(\pi\frac{r-R_{C_2}}{R_c-R_{C_2}}\right)+\frac{1}{2} & ,R_{C_2}\le r<R_C\\ 0 & ,r>R_C\end{cases}$$

这里，R_{C_2} 是一个平滑的截断参数，它允许在由 R_C 定义的局部区域的边界上平滑地将 $\mathbf{r}_i$ 的分量减小到零。这个平滑函数比之前使用的 R_{C_2} 更复杂一些。$S(\mathbf{r}_{ji})$ 减小了远离中心原子 i 的原子的权重。然后，我们定义径向函数 $g_M(s)$ 为深度势能切比雪夫特征中的切比雪夫多项式 C_M：

$$g_M(s)=C_M(2R_{\min}S-1)$$

这里，$R_{\min}$ 是最小 r 的输入。

为了构造这样的特征，我们首先计算四个分量的向量：

$$T_M(k)=\sum_{\mathbf{r}_{ji}<R_C}\hat{X}_{ji}(k)S(\mathbf{r}_{ji})g_M(S(\mathbf{r}_{ji}))$$

这里，$k=0,1,2,3$（四分量向量）。它们是由通常的 x,y,z 分量构成的，再加上 S 分量：

$$\{x_{ji},y_{ji},z_{ji}\}\to\{S(\mathbf{r}_{ji}),\hat{x}_{ji},\hat{y}_{ji},\hat{z}_{ji}\}$$

其中 $\hat{x}_{ji}=\frac{x_{ji}}{r_{ji}},\hat{y}_{ji}=\frac{y_{ji}}{r_{ji}},\hat{z}_{ji}=\frac{z_{ji}}{r_{ji}}$ 是 $\mathbf{r}_{ji}$ 的单位向量。

从这些 4D 向量中，我们可以缩并分量索引以得到标量特征：

$$F(M_1, M_2) = \sum_{k=0}^{3} T_{M_1}(k) T_{M_2}(k)$$

这里，M_1 也编码了除切比雪夫外的原子类型的数量。因此，如果最大切比雪夫阶数是 M，特征的数量是 $M \cdot n_{\text{type}} \cdot (M \cdot n_{\text{type}} + 1)/2$。我们可以通过设置不同的 M 来产生不同的特征。

5.5.7 DP-Gaussian (feature 8)

这个特征类似于 DP-Chebyshev，但我们使用高斯函数代替切比雪夫多项式，并且位置和宽度参数由用户指定。

类似于 DP-Chebyshev，4D 向量构造如下：

$$T_M(k) = \sum_{\mathbf{r}_{ji} < R_C} \hat{X}_{ji}(k) g_M(\mathbf{r}_{ji})$$

$$\hat{X}(0) = S(\mathbf{r}'), \quad \hat{X}(1) = \frac{x}{r}, \quad \hat{X}(2) = \frac{y}{r}, \quad \hat{X}(3) = \frac{z}{r}$$

$$g_M(\mathbf{r}) = f_C(\mathbf{r}) \cdot \exp\left(-\frac{(r - r_M)}{\omega_M}\right)$$

$$f_C(\mathbf{r}) = \frac{1}{2} \cos\left(\frac{\pi r}{R_C}\right) + \frac{1}{2}$$

缩并过程如下：

$$F(M_1, M_2) = \sum_{k=0}^{3} T_{M_1}(k) T_{M_2}(k)$$

这里，M_1 也编码了除切比雪夫外的原子类型的数量。因此，如果最大切比雪夫阶数是 M，特征的数量是 $M \cdot n_{\text{type}} \cdot (M \cdot n_{\text{type}} + 1)/2$。我们可以通过设置不同的 M 来产生不同的特征。

5.6 NN 操作演示

这里，我们以 MatPL [源码根目录/example/Cu/nn_demo] 为例，演示 NN 模型的训练、测试、lammps 模拟以及其他功能。案例目录结构如下所示。

```
Cu/nn_demo
    ├── nn_test.json
    ├── nn_train.json
    ├── train.job
```

```
└── nn_lmps/
    ├── in.lammps
    ├── lmp.config
    ├── forcefield.ff
    └── runcpu.job
```

- nn_train.json 是训练 NN 力场输入参数文件
- nn_test.json 是测试 NN 力场输入参数文件
- train.job 是 slurm 提交训练任务例子
- nn_lmps 目录下 为 NN 力场的 lammps md 例子
  - 力场文件 forcefield.ff
  - 初始结构 lmp.config
  - 控制文件 in.lammps
  - runcpu.job slurm 脚本例子

5.6.1 train 训练

在 nn_demo 目录下使用如下命令即可开始训练：

```
MatPL train nn_train.json
# 或修改环境变量之后通过 slurm 提交训练任务 sbatch train.job
```

输入文件解释 nn_train.json 中的内容如下所示，关于 NN 的参数解释，请参考 NN 参数手册：

```
{
    "format":"pwmat/movement",
    "train_data":["0_300_MOVEMENT", "1_500_MOVEMENT"],
    "valid_data":["valid_movement"],
    "model_type": "NN",
    "atom_type":[29]
}
```

训练结束后的力场文件目录请参考 model_record 详解

5.6.2 test 测试

```
MatPL test nn_test.json
```

test.json 中的内容如下所示，参数解释请参考 参数手册

```
{
    "format":"pwmat/movement",
    "test_data":["0_300_MOVEMENT", "1_500_MOVEMENT"],
    "model_type": "NN",
```

```
    "model_load_file":"./model_record/nn_model.ckpt"
}
```

测试结束后的力场文件目录请参考 test_result 详解

5.6.3 extract_ff

在训练结束后的 model_record 目录下，提取 nn_model.ckpt 文件

```
# 提取 nn 力场模型
MatPL extract_ff nn_model.ckpt
```

力场提取后，将得到一个 forcefield 目录，目录结构如下所示，该目录下的其他文件为描述符相关信息。

```
forcefield
  ├──fread_dfeat/
  ├──input/
  ├──output/
  └──forcefield.ff
```

5.6.4 lammps MD

NN 模型的 lammps 接口请参考 lammps-fortran 安装和使用。

将训练完成后生成的 `*.ff` 力场文件用于 lammps 模拟。lammps 的输入文件中设置以下内容:

```
pair_style      matpl
pair_coeff      * * 3 1 forcefield.ff 29
```

其中 3 表示使用 NN 模型产生的力场，1 表示读取 1 个力场文件，`forcefield.ff` 为 MatPL 生成的力场文件名称，29 为 铜 原子序数

5.7 LINEAR 模型

操作演示

5.7.1 模型介绍

LINEAR 模型特征值与 NN 模型中相同，区别是 NN 中特种值经过神经网络拟合出力场，而这里使用的是线性模型拟合。

5.8 LINEAR 操作演示

这里，我们以 MatPL [源码根目录/example/SiC] 为例，演示 Linear 模型的训练、测试、lammps 模拟以及其他功能。案例目录结构如下所示。

```
SiC
   ├── 1_300_MOVEMENT
   ├── 2_300_MOVEMENT
   ├── atom.config
   ├── linear_test.json
   ├── linear_train.json
   ├── train.job
   ├── linear_lmps
   └── MD
       └── MOVEMENT
```

- linear_train.json 是训练 linear 力场输入参数文件
- linear_test.json 是测试 linear 力场输入参数文件
- train.job 是 slurm 提交训练任务例子
- linear_lmps 目录下 为 linear 力场的 lammps md 例子
  - 力场文件 forcefield.ff
  - 初始结构 lmp.config
  - 控制文件 in.lammps
  - runcpu.job slurm 脚本例子

5.8.1 train 训练

在 SiC 目录下使用如下命令即可开始训练：**etot.input** 输入文件示例：

```
MatPL train linear_train.json
# 或修改环境变量之后通过 slurm 提交训练任务 sbatch train.job
```

输入文件解释

linear_train.json 中的内容如下所示，关于 Linear 的参数解释，请参考 参数手册：

```
{
    "train_data":["./1_300_MOVEMENT", "./2_300_MOVEMENT"],
    "valid_data":["./1_300_MOVEMENT"],
    "format":"pwmat/movement",
    "model_type": "Linear",
    "atom_type":[14, 6]
}
```

程序运行后，会在程序执行目录下生成 `forcefield` 目录:

```
forcefield
├── forcefield.ff
├── fread_dfeat
│   ├── energyL*
│   ├── forceL*
│   ├── linear*
│   ├── weight_feat.*
│   ├── energyL*
│   ├── ...
│   ├── feat*
│   ├── weight_feat.*
│   └── linear_fitB.ntype
├── input
│   └── *feature.in
└── (output)
    └── grid*    # feature 1, 2 时使用
```

5.8.2 test 测试

```
MatPL test linear_test.json
```

test.json 中的内容如下所示，参数解释请参考 参数手册

```
{
    "test_movement_file":["./MD/MOVEMENT"],
    "format": "pwmat/movement",
    "model_type": "Linear",
    "atom_type":[14, 6]
}
```

注意，这里 MOVEMENT 文件必须放在名为 MD 的目录下面，否则无法识别。测试结束后的力场文件目录 test_result 内容如下:

5.8.3 lammps MD

Linear 模型的 lammps 接口请参考 lammps-fortran 安装和使用。

将训练完成后生成的 `*.ff` 力场文件用于 lammps 模拟。lammps 的输入文件中设置以下内容:

```
pair_style      matpl
pair_coeff      * * 1 1 forcefield.ff 14 6
```

其中第一个 1 表示使用 Linear 模型产生的力场，第二个 1 表示读取 1 个力场文件，`forcefield.ff` 为 MatPL 生成的力场文件名称，14 和 6 分别为 Si 和 C 的原子序数

6 PWact 主动学习

6.1 PWact 平台介绍

机器学习力场相比于传统方法，能够更快和精确地预测材料性质和反应机理，当前最先进的基于深度学习的分子动力学已经能够做到百亿原子体系的模拟。但是由于机器学习方法的插值特性，对于训练集之外的相空间，MLFF 很难做出准确预测。由于训练数据通常是使用昂贵的第一性原理计算生成的，现实中很难获取到大量的从头算数据集，生成具有足够代表性的训练数据但不依赖大量从头算数据，对于提升模型的外推能力至关重要。PWact (Active learning based on PWmat Machine Learning Force Field) 是开源的基于 MatPL 的一套自动化主动学习平台，用于高效的数据采样。

PWact 平台包含主任务和任务分发器两部分，如 结构图 所示。

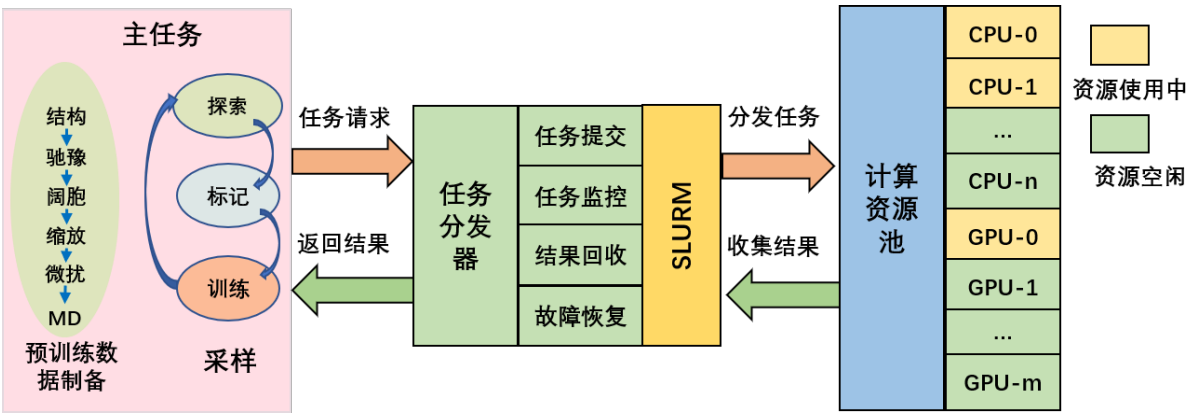


Figure 12: PWact 平台架构

主任务包括 预训练数据制备 以及 主动学习 两个模块。负责预训练数据制备和主动学习过程中的计算任务生成、以及结果收集。任务分发器接收到任务调度请求之后，根据计算资源使用状态以及任务申请资源情况将任务调度到对应的计算节点上，待任务执行完毕之后，收集计算节点的执行结果返回给主任务程序。

6.1.1 预训练数据制备

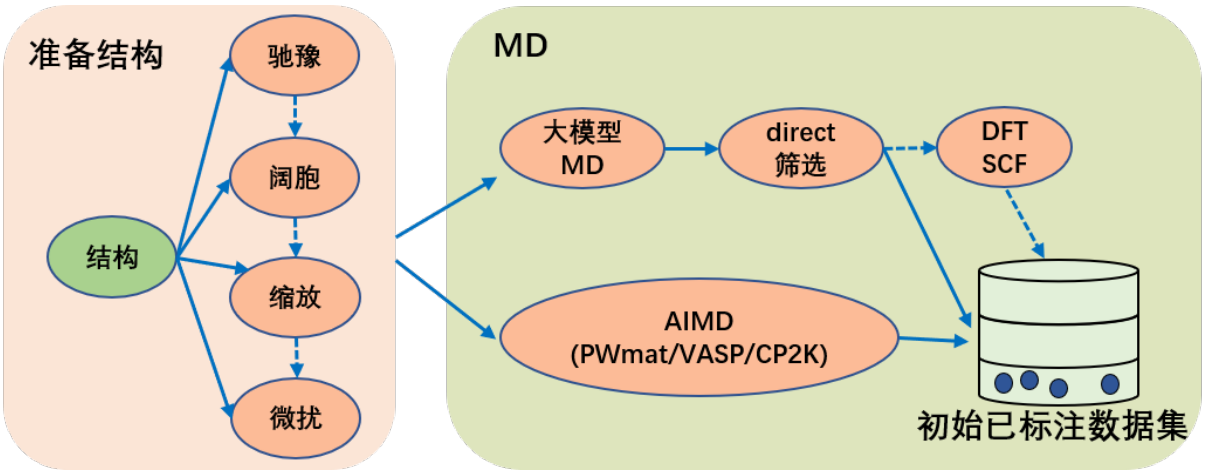


Figure 13: 预训练数据制备流程

预训练数据制备流程包括驰豫（支持 PWmat、VASP、CP2K）、阔胞、缩放晶格、微扰以及运行 MD 四个模块，并支持对这些模块的组合使用。对于 MD，（支持 PWmat、VASP、CP2K）的 AIMD 作为预训练集。同时，也支持使用大模型运行 MD，之后通过 direct 采样过滤掉相似构型后作为预训练集，这里同样支持使用 PWmat、VASP、CP2K 对过滤后的结构做标注（自洽计算获取能量和受力）。

- direct 采样：Qi J, Ko T W, Wood B C, et al. Robust training of machine learning interatomic potentials with dimensionality reduction and stratified sampling[J]. npj Computational Materials, 2024, 10(1): 43.

6.1.2 主动学习

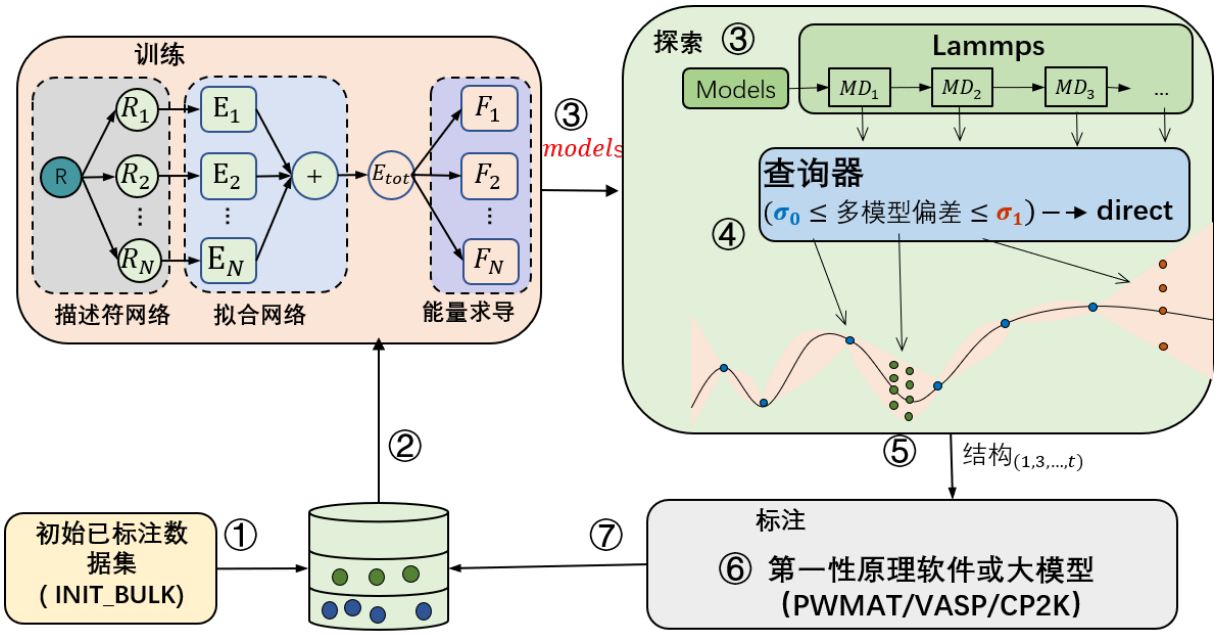


Figure 14: 主动学习流程

主动学习流程包括训练、构型探索以及标注模块。首先，训练模块做模型训练；之后将训练好的模型送入探索模块。探索模块调用力场模型做分子动力学模拟，模拟结束后把得到的分子运动轨迹送入查询器做不确定性度量；查询完成后，把待标注构型点送入标注模块；最后标注模块做自洽计算，得到能量和力，作为标签与对应构型一起送入已标注数据库中；重复上述步骤，直到收敛。

- 对于模型训练，我们这里支持 MatPL 中的 DP model、DP model with compress 以及 DP model with type embedding，以及 NEP 模型。
- 对于不确定性度量，提供了常用了基于多模型委员会查询的方法，并且也提供了我们最新设计的 单模型的基于卡尔曼滤波的不确定性度量方法 KPU (Kalman Prediction Uncertainty, KPU)。该方法能够在接近委员会查询精度的情况下，将模型训练的计算开销减少到 1/N, N 为委员会查询模型数量，欢迎用户尝试。对于 KPU 方法，仅适用于 DP 模型。
- 【可选项】通过 direct 采样方法 降低候选集中的相似结构数量。
- 对于标注，支持 PWmat、VASP、CP2K 或 使用大模型推理。

6.1.3 依赖应用

- PWact 作业调度采用 SLURM 集群管理和作业调度系统，需要您的计算集群上已安装 SLURM。
- PWact 的 DFT 计算支持 PWmat 、VASP、CP2K，需要您的计算机群已安装 PWMAT、VASP 或 CP2K。
- PWact 使用的力场模型使用 MatPL训练, MatPL 安装方式参考 MatPL 在线手册。

- PWact Lammps 分子动力学模拟 基于 lammps-MatPL，安装方式参考 MatPL 在线手册。
- Direct 采样和大模型采样，目前在 龙讯超算云(Mcloud) 做了预装，支持 direct 采样和 eqv2 大模型。对于用户自己的环境，需要自己安装相应环境，并提供处理脚本。

6.1.4 安装流程

PWact 支持 pip 命令安装与源码安装两种安装方式。

6.1.4.1 pip 命令安装 安装包已上传至 pypi 官网，支持直接使用 pip install 安装。

```
pip install pwact

# 安装 pwact, 如果已安装，则升级到最新版本
pip install pwact --upgrade

# 列出所有可安装版本
pip index versions pwact
# 输出结果示例:
# pwact (0.1.27)
# Available versions: 0.1.27, 0.1.25, 0.1.23, 0.1.10
# 安装指定版本
pip install pwact==n.m.o
```

6.1.4.2 github 源码安装 源码安装适用于需要修改源码的用户，否则建议您通过 pip 安装 即可。源码下载

```
git clone https://github.com/LonxunQuantum/PWact.git
```

或者

```
git clone https://gitee.com/pfsuo/pwact.git
```

gitee 更新可能没有 github 及时，建议优先从 github 下载

源码下载后，进入源码的根目录（与 setup.py 同一级）执行命令

```
pip install .
# 或者加开发者选项， 安装时不复制文件，而是直接从源码文件读取，任何对源码的修改都会立即生效
# pip install -e .
# 从源码安装执行完毕后，需要把 pwact 加入环境变量
# export PYTHONPATH=/the/path/pwact:$PYTHONPATH
```

PWact 开发语言采用 Python，支持 Python 3.9 以及以上的版本。建议用户直接使用 MatPL 的 Python 运行环境 即可。

如果您需要为 PWact 单独创建虚拟环境，只需要安装以下依赖包即可（与您的 Python 版本相匹配，支持 Python 3.9 以及以上）。

```
pip install numpy pandas tqdm pwdata
```

6.1.5 命令列表

PWact起始命令为 pwact，包括一系列功能命令，所有功能命令都对大小写不敏感。例如 INIT_BULK、Init_BUlk 或 init_bulk 都是等价的。

6.1.5.1 输出可用命令列表

```
pwact  [ -h / --help / help ]

# 输出具体命令 cmd_name 对应的参数列表
pwact cmd_name -h
```

6.1.5.2 初始训练集制备

```
pwact init_bulk param.json resource.json
```

6.1.5.3 主动学习

```
pwact run param.json resource.json
```

对于上述两个命令，json 文件名称可以用户修改，但是要求 param.json 和 resouce.json 的输入顺序不能变。

6.1.5.4 gather_pwdata 命令 搜索主动学习目录下所有探索到的所有结构，并将结果保存到 final_pwdata 目录下，数据格式与主动学习中设置的 'data_format' 格式相同。目录结构如下所示，final_pwdata_list.txt 保存了当前目录下每个 iter 数据的目录。

```
final_pwdata/
final_pwdata_list.txt  iter.0000  iter.0001  iter.0002 ...
```

```
pwact gather_pwdata -i .
```

这里 -i 指定主动学习的根目录所在路径（案例中对应 run_iter 目录）。

6.1.5.5 kill 命令 结束正在执行的 init_bulk 任务，如 驰豫（relax）、AIMD 任务

```
# 进入到执行 pwact init_bulk 命令时的所在目录
pwact kill init_bulk
```

结束正在执行的 run 任务，包括正在运行的训练、探索（MD）或者标记任务

```
# 进入到执行 pwact run 命令时的所在目录
pwact kill run
```

上面的 kill 命令功能您也可以通过手动操作替代，需要您第一步，结束正在执行的主进程，即执行 pwact init_bulk 或 pwact run 的窗口；第二步，需要您手动结束正在执行的 slurm 任务。

考虑到手动操作可能会误结束您的其他进程，建议您使用命令结束。

使用命令结束进程后，建议您查看命令输出信息，并使用 slurm 命令查看是否有未结束的进程。

6.1.5.6 filter 命令 测试指定上、下限设置下的选点情况

```
pwact filter -i iter.0000/explore/md -l 0.01 -u 0.02 -s
```

该命令将检测位于 iter.0000/explore/md 目录下（该轮次探索到的所有轨迹）使用下限 0.01 和上限 0.02 的选点结果 (如下面的例子所示)，-s 为可选项，用于指定是否将详细的选点信息做保存。

```
Image select result (lower 0.01 upper 0.02):
 Total structures 972    accurate 20 rate 2.06%    selected 44 rate
 ↪  4.53%    error 908 rate 93.42%

Select by model deviation force:
Accurate configurations: 20, details in file accurate.csv
Candidate configurations: 44
        Select details in file candidate.csv
Error configurations: 908, details in file fail.csv
```

6.1.6 输入文件解读

PWact 包括两个输入文件 `param.json` 和 `resource.json`，用于初始数据集制备或者主动学习。PWact 对于两个 JSON 文件中键的大小写输入不敏感。

6.1.6.1 输入文件 param.json 初始训练集制备 init_param.josn

对构型（VASP、PWmat 格式）进行驰豫、扩胞、缩放、微扰和 AIMD（PWMAT、VASP、CP2K）设置。

主动学习 run_param.josn

主动学习流程中的训练设置（网络结构、优化器）、探索设置（lammps 设置、选点策略）以及标记设置 (VASP/PWmat 自洽计算设置)。

6.1.6.2 输入文件 resource.json resource.json

计算集群资源设置，包括对训练、分子动力学（MD）、DFT 计算（SCF、Relax、AIMD）使用的计算节点、CPU、GPU 资源设置，以及对应的运行软件（Lammps、VASP、PWMAT、MatPL）。

6.1.7 主动学习案例

- 硅的主动学习
- 硅的主动学习 (bigmodel+direct)
- 金银合金的主动学习

6.2 init_bulk param.json 设置

init_bulk 初始训练集制备，包含对构型（VASP、PWmat 等格式）进行驰豫、阔胞、缩放、微扰和 AIMD (支持 DFTB、PWMAT、VASP）设置。参数列表如下。

6.2.1 data_format

用于设置 init_bulk 执行结束后的得到的数据格式，默认为扩展的 xyz 格式 `extxyz`。

6.2.2 reserve_work

是否保留临时工作目录，默认值为 `false`，不保存。

6.2.3 interval

用于设置提取数据时，从轨迹中选取结构的间隔，即在轨迹中，每隔多少个结构选取一个构型，默认值为 `1`。

6.2.4 sys_config_prefix

用于设置初始构型的路径前缀，可选参数，与 `sys_configs/config` 配合设置。可以是绝对路径或者相对路径，相对路径为当前目录。

例子:`"sys_config_prefix":"/data/structure","config":"atom.config"`, 则 `atom.config` 的实际路径是 `/data/structure/atom.config`

6.2.5 sys_configs

设置构型的文件路径、驰豫（relax)、阔胞（super cell)、缩放晶格（scale)、微扰原子位置（pertub)、AIMD。完整的参数如下例所示。

6.2.5.1 config 设置构型的文件路径，如果设置了 `sys_config_prefix` 则进行路径拼接，否则使用 config 中设置的路径作为 config 路径。

6.2.5.2 format 设置构型的文件类型，支持 VASP 的 POSCAR 或者 PWMAT 的 `atom.config` 格式。如果是 POSCAR 文件，则值为 `"vasp/poscar"`, 默认值为 `"pwmat/config"`。

6.2.5.3 relax 是否对 config 做驰豫，默认值为 `"true"`。

6.2.5.4 relax_input_idx 设置驰豫使用的控制文件，与 relax_input 配合使用，指定控制文件的位置，如 例子 中所示，使用 `relax_input` 中设置的 `relax_etot1.input` 文件作为 PWMAT 控制文件。默认值为 0，即使用 `relax_input` 中的第一个文件作为控制文件。

6.2.5.5 super_cell 用于阔胞设置，可选参数，如不设置，则对结构不做阔胞。数据格式为 list，支持如下格式输入：`[1, 1, 2]` 或 `[[1,0,0],[0, 2, 0],[0,0,1]]` 或 `[1,0,0,0, 2, 0,0,0,1]`。

6.2.5.6 scale 用于对晶格的缩放设置，可选参数，如不设置，则对结构不做缩放。数据格式为 list，如 `[0.9, 0.95, 0.96, 0.97]`，表示对结构晶格分别进行 0.9, 0.95，0.96, 0.97 微扰，将得到四个微扰后的结构。

6.2.5.7 perturb 对结构的原子位置做扰动,配合 `cell_pert_fraction`、`atom_pert_distance` 使用，`perturb` 值为扰动后生成的结构数量。可选参数，不设置则不做扰动。

6.2.5.8 cell_pert_fraction 对晶格的扰动。对 9 个晶格值分别加上从 [-cell_pert_fraction, cell_pert_fraction] 范围内的均匀分布中随机采样的值，默认值为 `0.03`。

6.2.5.9 atom_pert_distance 原子坐标的扰动（Angstrom）。对每个原子的三个坐标值，分别加上从 [-atom_pert_distance，atom_pert_distance] 范围内的均匀分布中随机采样的值。默认值为 `0.01`。

6.2.5.10 aimd 是否对结构做分子动力学模拟，默认值为 true。

6.2.5.11 aimd_input_idx 设置 AIMD 使用的控制文件，与 aimd_input 配合使用，指定控制文件的位置，如 例子 中所示，使用 `aimd_input` 中设置的 `aimd_etot.input` 文件作为 PWMAT 控制文件。默认值为 0，即 使用 `aimd_input` 中的第一个文件作为控制文件。

6.2.5.12 bigmodel 是否使用大模型运行 MD，默认值为 False

6.2.5.13 bigmodel_input_idx 设置 bigmodel 使用的脚本，与 bigmodel_input 配合使用，指定脚本文件的位置。

6.2.5.14 direct 设置 是否使用 direct 方法筛选结构，默认值 False，与 direct_input 配合使用，direct 为 True 时，必须在 direct_input 中指定 direct 的 脚本文件所在位置。

6.2.6 sys_config 设置例子

```
    "sys_config_prefix": "../../si_example/init_bulk",
    "sys_configs": [{"config":"./structures/49.config",
                     "relax":true,
                     "_relax_input_idx":0,
                     "super_cell":[1, 1, 2],
                     "scale":[0.9,0.95],
                     "perturb":3,
                     "cell_pert_fraction":0.03,
                     "atom_pert_distance":0.01,
                     "aimd":true,
                     "_aimd_input_idx":0
                     },
                     {"config":"./structures/44_POSCAR",
                     "format":"vasp/poscar",
                     "relax":false,
                     "super_cell":[[1,0,0],[0, 2, 0],[0,0,1]],
                     "perturb":2,
                     "aimd":true,
                     "aimd_input_idx": 1
                     }
        ]
```

这里设置了 49.config 和 44_POSCAR 两个结构，分别是 pwmat/config （默认格式）和 vasp/poscar 格式。- 对 49.config 操作如下：step1. 对 49.config 做 relax，使用的 relax 控制文件为 relax_input 中的第一个文件；step2. 对 relax 得到的结构，分别对晶格做 0.9, 0.95 缩放，缩放方式参考；step3. 对缩放后得到两个文件做晶格和原子位置微扰，各自微扰出 3 个结构，微扰方式参考；step4. 对微扰后得到的 6 个结构做 AIMD，使用的 md 控制文件为 aimd_input 中设置的第 1 个文件。init_bulk 执行结束后将得到 6 条 AIMD 轨迹。- 对 44_POSCAR 操作如下：step1. 对 49.config 做阔胞，按照 [1,0,0],[0, 2, 0],[0,0,1] 阔胞，阔胞方式参考；step2. 对阔胞后的结构做微扰；step3. 对微扰后得到的 2 个结构做 AIMD，使用的 md 控制文件为 aimd_input 中设置的第 2 个文件。init_bulk 执行结束后将得到 2 条 AIMD 轨迹。

因此 init_bulk 执行结束后将得到 6 条轨迹，之后会自动将轨迹提取为 data_format 中指定的文件格式。

6.2.7 dft_style

设置 Relax 和 AIMD 使用哪种 DFT 计算软件，默认值为 pwmat, 也支持 VASP 格式，如果是 VASP 格式，则设置为 vasp。

6.2.8 pseudo

设置 PWMAT 或 VASP 赝势文件所在路径，为 list 格式，赝势文件路径可以为绝对路径或相对路径（相对于当前路径）。

6.2.9 gaussian_param

CP2K 或 PWMAT 高斯基组参数设置，basis_set_file 和 potential_file 指定基组和势函数文件路径。atom_list, basis_set_list, potential_list 配合使用，分别指定元素对应的基组和势函数设置。kspacing 用于设置 K 点，用法与 PWMAT KSPACKING 设置 相同。

```
"gaussian_param": {
    "basis_set_file":"./init_bulk/BASIS_MOLOPT_1",
    "potential_file":"./init_bulk/POTENTIAL_1",
    "atom_list":["Si"],
    "kspacing" :0.4,
    "basis_set_list":["SZV-MOLOPT-SR-GTH"],
    "potential_list":["GTH-PBE-q4"]
}
```

6.2.10 relax_input

设置 Relax 的 输入控制文件。如果存在多个 relax 控制文件，则按照 list 格式组织。详细的设置请参考下面的例子。

6.2.11 aimd_input

设置 AIMD 的 输入控制文件。如果存在多个 aimd 控制文件，则按照 list 格式组织，对于只是用单个文件的情况，也可以设置为 dict 格式。

6.2.11.1 input 设置输入控制文件的路径，可以为绝对路径或相对路径（相对于当前路径）。

6.2.11.2 kspacing 该参数为 PWMAT 的输入参数，用于设置 K 点，可选参数。如果在 etot.input 文件中未设置 MP_N123 参数，则使用该参数设置 K 点。如果文件中已经设置了 MP_N123，则对该参数的设置会造成错误，请确保 MP_N123 与 kspacing 只能同时存在一个。该参数为 PWMAT 的输入参数，用于设置 K 点，可选参数。如果在 etot.input 文件中未设置 MP_N123 参数，则使用该参数设置 K 点。

如果 etot.input 文件中未设置 MP_N123，且 kspacing 未设置，则采用默认设置 kspacing 值为 0.5。

注意，不能同时设置 MP_N123 与 kspacing。

6.2.11.3 flag_symm 该参数为 PWMAT 的输入参数，用于设置 K 点，可选参数。对于 Relax 或者 SCF 计算，默认值为 0, 对于 AIMD 计算，默认值为 3。

6.2.12 bigmodel_input

设置 大模型的脚本文件。如果存在多个脚本文件，则按照 list 格式组织。

对大模型的接口设置，请参考例子 si_direct_bigmodel。

6.2.13 direct_input

设置 direct 采样的脚本文件，为单文件路径。

对 direct 的接口设置，请参考例子 si_direct_bigmodel。

6.2.14 完整例子

```
    {
        "reserve_work": true,
        "sys_configs":[{
            "config":"atom.config",
            "relax":true,
            "relax_input_idx":1,
            "super_cell":[1, 1, 2],
            "scale":[0.9,0.95],
            "perturb":20,
            "cell_pert_fraction":0.03,
            "atom_pert_distance":0.01,
            "aimd":true,
            "aimd_input_idx":0
        }],

    "dft_style":"PWmat",
```

```
    "pseudo" : ["/path/Si.SG15.PBE.UPF"],

    "relax_input":[{
            "input":"relax_etot.input",
            "kspacing":0.5,
            "flag_symm":"0"
        },{
            "input":"relax_etot1.input",
            "kspacing":0.3,
            "flag_symm":"0"
        },{
            "input":"relax_etot2.input",
            "kspacing":0.4,
            "flag_symm":"0",
            "_flag":"1 个整数, relax or scf 0 , aimd 3, 磁性体系 2"
        }],

    "aimd_input":[{
            "input":"aimd_etot.input",
            "kspacing":0.5,
            "flag_symm":"3"
        },{
            "input":"aimd_etot1.input",
            "kspacing":0.5,
            "flag_symm":"3"
        }]
    }
```

上述参数作用 - 对 atom.config 做 relax；- 对 relax 后的结构做阔胞（1,1,2)；- 对阔胞后的结构对晶格分别进行 0.9、0.95 缩放; - 对缩放后得到两个结构的原子位置微扰，各自扰动生成 20 个结构；- 对扰动后得到的 40 个结构做 AIMD 模拟；- 将 AIMD 轨迹自动提取为 PWdata 数据格式作为预训练数据。

6.2.14.1 relax_input 和 aimd_input kspacing和flag_symm是 PWMAT 的 K 点设置，如果没有在 etot.input 文件中设置 MP_N123 参数，那么程序会使用这两个参数设置 K 点。因此，如果您已经 etot.input 设置了 MP_N123，那么，您可以将relax_input 和 aimd_input 简写为如下形式

```
    "relax_input":[
        "relax_etot.input",
        "relax_etot1.input",
        "relax_etot2.input"
    ],

    "aimd_input":[
        "aimd_etot.input"
```

```
        "aimd_etot1.input"
    ]
```

如果您对在 sys_configs 中的所有结构，都使用同一个 relax 或者 aimd，那么您可以进一步将参数简写为

```
    "relax_input":"relax_etot.input",
    "aimd_input": "aimd_etot.input"
```

此时，在 sys_configs 您不必再写 relax_input_idx 和 aimd_input_idx 参数，此时您的 sys_configs可以写为如下所示。

```
    "sys_configs":[{
        "config":"atom.config",
        "relax":true,
        "super_cell":[1, 1, 2],
        "scale":[0.9,0.95],
        "perturb":20,
        "cell_pert_fraction":0.03,
        "atom_pert_distance":0.01,
        "aimd":true
    }]
```

6.2.14.2 pwmat 设置例子

pwmat K 点通过在控制文件中设置 MP_N123，例子：

```
    "dft_style":"PWmat",
    "relax_input":["relax-agau-etot.input", "relax_ag-etot.input",
     ↪  "relax_au-etot.input"],
    "aimd_input": ["aimd_agau-etot.input",   "aimd_ag-etot.input",
     ↪  "aimd_au-etot.input" ],
    "pseudo" : ["../Ag.SG15.PBE.UPF", "../Au.SG15.PBE.UPF"]
```

pwmat K 点通过 kspacing 设置，例子：

```
    "dft_style":"PWmat",
    "relax_input":[{
            "input":"relax_etot0.input",
            "kspacing":0.5,
            "flag_symm":"0"
        },{
            "input":"relax_etot1.input",
            "kspacing":0.3,
            "flag_symm":"0"
        },{
            "input":"relax_etot2.input",
            "kspacing":0.4,
```

```
            "flag_symm":"0",
            "_flag":"1 个整数, relax or scf 0 , aimd 3, 磁性体系 2"
        }],

    "aimd_input":[{
            "input":"aimd_etot0.input",
            "kspacing":0.5,
            "flag_symm":"3"
        },{
            "input":"aimd_etot1.input",
            "kspacing":0.5,
            "flag_symm":"3"
        }],
    "pseudo" : ["../Ag.SG15.PBE.UPF", "../Au.SG15.PBE.UPF"]
```

pwmat gaussian 基组设置例子:

```
    "dft_style":"PWmat",
    "relax_input":["relax_etot.input",
     ↪  "relax_etot1.input","relax_etot2.input"],
    "aimd_input": ["aimd_etot1.input", "aimd_etot2.input"],
    "gaussian_param": {
        "basis_set_file":"./BASIS_MOLOPT_1",
        "potential_file":"./POTENTIAL_1",
        "atom_list":["Si"],
        "basis_set_list":["SZV-MOLOPT-SR-GTH"],
        "potential_list":["GTH-PBE-q4"]
    }
```

6.2.14.3 vasp 设置例子

```
    "dft_style":"vasp",
    "relax_input":["INCAR_relax_AgAu", "INCAR_relax_Ag",
     ↪  "INCAR_relax_Au"],
    "aimd_input": ["INCAR_md_AgAu",   "INCAR_md_Ag", "INCAR_md_Au" ],
    "pseudo" : ["../Ag_POTCAR", "../Au_POTCAR"]
```

6.2.14.4 cp2k 设置例子

```
    "dft_style":"cp2k",
    "gaussian_param": {
        "basis_set_file":"../BASIS_MOLOPT_1",
        "potential_file":"../POTENTIAL_1",
        "atom_list":["Ag", "Au"],
```

```
        "basis_set_list":["SZV-MOLOPT-SR-GTH-q11",
         ↪  "SZV-MOLOPT-SR-GTH-q11"],
        "potential_list":["GTH-PBE", "GTH-PBE"],
        "kspacing":0.5
    },
    "aimd_input":["aimd_cp2k_AgAu.inp", "aimd_cp2k_Ag.inp",
     ↪  "aimd_cp2k_Au.inp"],
    "relax_input":["relax_cp2k_AgAu.inp", "relax_cp2k_Ag.inp",
     ↪  "relax_cp2k_Au.inp"]
```

6.3 run param.json 设置

主动学习流程包含训练设置（网络结构、优化器）、探索设置（lammps 设置、选点策略）以及标记设置(VASP/PWmat 自洽计算设置)。参数列表如下所示

6.3.1 reserve_work

是否保留临时工作目录，默认值为 `false`，每轮次主动学习执行结束之后，自动删除临时工作目录。

6.3.2 reserve_md_traj

是否保留 md 运行轨迹，默认值为 `false`，每轮次主动学习执行结束之后，自动删除 md 运行轨迹文件。

6.3.3 reserve_scf_files

是否保留自洽计算的所有结果文件，默认值为 `false`，设置为 `false` 之后，每轮次主动学习结束之后，对于 PWMAT 自洽计算，只保留 REPORT, `etot.input`,`OUT.MLMD`, `atom.config` 四个文件，对于 VASP 只保留 OUTCAR, POSCAR, INCAR 三个文件。

6.3.4 data_format

用于设置主动学习中初始训练集、采集到的数据集格式，默认为扩展的 xyz 格式 `extxyz`。

6.3.5 init_data

初始训练集所在目录，为 list 格式。可以是绝对路径或者相对路径（当前目录)。

6.3.6 valid_data

验证集所在目录，为 list 格式，可以是绝对路径或者相对路径（当前目录）。如不设置，则在主动学习训练模型过程中不输出验证集结果。

6.3.7 init_model_list

用于设置初始探索模型，如果已有 MatPL 训练的 DP 或者 NEP 力场，并且希望从这些力场开始探索工作，则将力场文件路径写入 init_model_list 即可。

注意，这里要求力场数量要与 strategy/model_num 中一致，模型类型要与 `train/model_type` 中的模型类型一致。并且模型的训练参数将会自动从力场文件中提取，在 `train_input_file` 或者 `train` 字典中设置的模型网络和描述符参数将失效。

6.3.8 use_pre_model

在当前步的探索中，使用上一步训练得到的力场，默认值为 true。

6.3.9 train

模型训练参数，用于指定模型网络结构、优化器。详细的参数设置参考 `MatPL 训练参数`。您可以像如下例子中所示，设置训练的全部参数，也可以使用单独的 json 文件，只需要在参数 `train_input_file` 中指定训练的 json 文件所在路径即可。

6.3.9.1 train_input_file 可选参数，如果您有单独的 MatPL 输入文件，您可以使用该参数指定文件所在路径。否则您需要设置如下例中所示参数。参数的详细解释您在可以在 MatPL 参数列表中查看。

```
    "train": {
        "model_type": "DP",
        "atom_type": [
            14
        ],
        "seed": 2023,
        "model": {
            "descriptor": {
                "Rmax": 6.0,
                "Rmin": 0.5,
                "M2": 16,
                "network_size": [25, 25, 25]
            },
            "fitting_net": {
                "network_size": [50, 50, 50, 1]
```

```
            }
        },
        "optimizer": {
            "optimizer": "LKF",
            "epochs": 30,
            "batch_size": 4,
            "print_freq": 10,
            "block_size": 5120,
            "kalman_lambda": 0.98,
            "kalman_nue": 0.9987,
            "train_energy": true,
            "train_force": true,
            "train_virial": false,
            "pre_fac_force": 2.0,
            "pre_fac_etot": 1.0,
            "pre_fac_virial": 1.0
        }
    }
```

由于 MatPL 中设置的默认参数已经能够支持大部分的训练需求，因此，您可以简写为如下形式，将采用标准的 DP 模型 使用 LKF 优化器训练。

```
"train": {
        "model_type": "DP",
        "atom_type": [14]
}
```

PWact 同时支持 MatPL 的 DP 和 NEP 力场。

6.3.10 strategy

用于设置主动学习的不确定性度量方法，以及是否采用模型压缩做加速。

6.3.10.1 uncertainty 用于设置不确定性度量策略，当前支持多模型委员会查询方法 (committee)。

6.3.10.2 lmps_tolerance 当 lammps 部分轨迹由于各种原因（如力场不准确导致丢失了原子、原子距离太近等）造成 MD 过程为正常执行结束时，是否终止主动学习过程。默认值为 true，即不终止。

6.3.10.3 lower_model_deiv_f 该参数用于设置偏差的下界，如果偏差值小于该下界，则认为模型对构型的预测准确，不需要标注，默认值为 0.05。

6.3.10.4 upper_model_deiv_f 该参数用于设置偏差的上界，如果偏差值大于该上界，则该构型本身不符合物理意义，不需要标注，默认值为 `0.15`。

这里使用的最大偏差值，计算公式如下所示：

$$\varepsilon_t = \max_i \left(\sqrt{\frac{\sum_1^w \left\| F_{w,i}(R_t) - \hat{F}_i \right\|^2}{W}} \right), \quad \hat{F}_i = \frac{\sum_1^W F_{w,i}}{W}$$

这里 W 为模型数量，i 为原子下标。

6.3.10.5 model_num 该参数用于设置使用 `committee` 方法作为不确定性度量时使用的模型数量，默认值为 `4`，该值的设置应该 >=3。

6.3.10.6 max_select 该参数用于设置一轮次主动学习中，对于未设置 `select_sys` 参数的每个初始探索结构对应最大选取构型用于标注的数量。当待标注结构数目超过该值时，将随机从待标注结构中选择 `max_select` 个结构做标注。默认不设置，即不做限制。

例如对于如下 md 探索设置，由于未设置 `select_sys`，如果设置了 `max_select`，则对于 `sys_idx` 中指定的这两个结构，分别最多采集 `max_select` 个结构，因此这里对于该 md 探索设置，最多采集 $2×max_select $ 个结构用于标记。

6.3.10.7 direct 该参数用于设置是否开启 direct 采样，用于对多模型偏差选出的结构进一步过滤，去除相似的结构。默认值为 False。

6.3.10.8 direct_script 该参数用于设置 direct 方法使用的处理脚本，direct 为 True 时，必须指定处理脚本。

对 direct 方法的接口设置，请参考例子 si_direct_bigmodel。

6.3.10.9 compress 该参数用于设置是否对模型做压缩，经过压缩后的模型精度会略有下降，但是模拟速度会有翻倍提升。默认值为 `false`，即不使用模型压缩。

6.3.10.10 compress_order 该参数用于设置模型压缩的方式，默认值为 `"compress_order":3`，即使用三阶多项式压缩。也可以设置为 `"compress_order":5`，即使用五阶多项式压缩，相比于三阶多项式精度会更高，但是速度比三阶稍慢。

6.3.10.11 Compress_dx 该参数用于设置模型压缩是的网格大小，默认值为 `"Compress_dx":0.01`。

6.3.10.12 例子 对于 committee 方式的选点策略

```
"strategy": {
    "uncertainty":"committee",
    "lower_model_deiv_f": 0.1,
    "upper_model_deiv_f": 0.2,
    "model_num": 4,
    "max_select": 50,
    "compress": false
}
```

如果您需要开启模型压缩，则

```
"strategy": {
    "uncertainty":"committee",
    "lower_model_deiv_f": 0.1,
    "upper_model_deiv_f": 0.2,
    "model_num": 4,
    "max_select": 50,
    "compress": true,
    "compress_order":3,
    "Compress_dx":0.01
}
```

6.3.11 explore

用于设置主动学习每个轮次探索过程的分子动力学设置

6.3.11.1 sys_config_prefix 用于设置待探索的结构文件路径前缀，可选参数，与 sys_configs 配合使用。可以是绝对路径或者相对路径，相对路径为当前目录。

6.3.11.2 sys_configs 设置待探索的结构文件路径，如果设置了 sys_config_prefix 则进行路径拼接，否则使用 sys_configs 中设置的路径作为 config 路径。

该参数为 list 格式，对于 PWMAT 格式结构文件，直接写出文件路径即可，对于 VASP 格式结构文件，需要设置 format 格式，如下例中所示。

```
    "sys_config_prefix": "../../structures",
    "sys_configs": [
                    {"config":"POSCAR", "format":"vasp/poscar"},
                    "49.config",
                    "45.config",
                    "41.config",
                    "37.config",
```

```
                "33.config",
                "29.config",
                "25.config",
                "21.config",
                "17.config",
                "1.config"
        ]
```

例如对于 49.config 文件，它的文件路径为../../structures/49.config。

6.3.11.3 lmps_prefix 可选参数，用于设置 lammps 探索的输入控制文件路径前缀，可选参数，与 lmps_in 配合使用。可以是绝对路径或者相对路径，相对路径为当前目录。

例子："lmps_prefix":"/data/in_lmps_files", "lmps_in":"in0.lmps"，则 in0.lmps 的实际路径是 /data/in_lmps_files/in0.lmps。

6.3.11.4 lmps_in 可选参数，设置设置 lammps 探索的输入控制文件路径，如果设置了 lmps_prefix 则进行路径拼接，否则使用 lmps_in 中设置的路径作为 lammps 控制文件的路径。

该参数为 list 格式，如下例中所示。

```
    "lmps_prefix": "../../in_lmps_files",
    "lmps_in": [
                "in0.lmps",
                "in1.lmps",
                "in2.lmps",
                "in3.lmps",
                "in4.lmps",
                "in5.lmps"
        ]
```

6.3.11.5 md_jobs 设置每个轮次的主动学习分子动力学参数。为 list 格式，第 i 个元组代表第 i 个轮次主动学习对应设置。每个元组内部可以为 dict 格式，或者 dict 格式的 list 数组。注意：以下 md 设置中，对于时间的单位都使用 units metal。

对于 lammps 的输入控制，pwact 提供了两种方式。第一种是在 param.json 中提供的关键字设置，控制探索需要的步数、以及 lammps 温度、压强、系综，可参考例子。第二种是通过用户提供的 lammps.in 文件控制，参数为 lmps_prefix 和 lmps_in。可参考 金银合金主动学习操作案例。

lmps_in 设置方式在 >= pwact-0.4 版本中开始支持。

对于 lmps_in，如果用户提供了 lammps.in 文件，pwact 在运行时，会自动维护 lammps.in 文件中的以下字段。

- “dump_freq”，该参数通过 trj_freq 设置，用于设置每隔多少步采样一次

- “units”、“boundary”、“atom_style”，由于 MatPL 力场只支持周期性的体系模拟，因此这三个关键字是固定格式，内容为

```
units           metal
boundary        p p p
atom_style      atomic
```

- “restart”，pwact 中自动设置每个 1 万步保存一次运行状态
- “read_data”，该参数用设置初始结构所在路径，在 pwact 在探索时该值会自动设置为所需路径
- “mass”、“pair_style”、“pair_coeff”，该三个参数用于设置机器学习的力场，以对硅元素体系的探索为例，在 pwact 在探索时该值会自动设置为：

```
mass    1    28.086
pair_style   matpl  0_torch_script_module.pt
↪  1_torch_script_module.pt 2_torch_script_module.pt
↪  3_torch_script_module.pt  out_freq ${DUMP_FREQ}  out_file
↪  model_devi.out
pair_coeff   * * 14
```

- “dump”，该值用于设置轨迹的保存格式，在 pwact 中该值会自动设置为如下内容，并插入到 lammps.in 文件中的第一个 run 命令所在行前面，这里 DUMP_FREQ 值为trj_freq参数中所设值：

```
dump 1 all custom ${DUMP_FREQ} traj/*.lammpstrj id type x y z
↪  fx fy fz
```

- “dump_modify”，如果用户设置了该值，pwact 在使用时会自动将该行的’dump_modify dump-ID’替换为‘dump_modify 1’，即设置 dump-ID 与 dump 中一致，其他不变

这里以如下设置为例，该 lmp.in 文件为硅的 lammps 模拟输入文件：

```
variable        NSTEPS          equal 400
variable        THERMO_FREQ     equal 5
variable        DUMP_FREQ       equal 5
variable        restart         equal 0
variable        TEMP            equal 500.000000
variable        PRESS           equal 100.000000
variable        TAU_T           equal 0.100000
variable        TAU_P           equal 0.500000

units           metal
boundary        p p p
atom_style      atomic

neighbor        1.0 bin
neigh_modify    delay 10
```

```
box               tilt large
if "${restart} > 0" then "read_restart lmps.restart.*" else
 ↪  "read_data lmp.config"
change_box        all triclinic

thermo_style    custom step temp pe ke etotal press vol lx ly lz xy xz
 ↪  yz
thermo          ${THERMO_FREQ}
restart         10000 lmps.restart

if "${restart} == 0" then "velocity        all create ${TEMP} 76752"
fix             1 all npt temp ${TEMP} ${TEMP} ${TAU_T} iso ${PRESS}
 ↪  ${PRESS} ${TAU_P}

timestep        0.001000
run             ${NSTEPS} upto
```

运行时，lmp.in 会替换为如下内容：

```
variable        DUMP_FREQ       equal 5
variable        restart         equal 0
units           metal
boundary        p p p
atom_style      atomic

if "${restart} > 0" then "read_restart lmps.restart.*" else
 ↪  "read_data lmp.config"

mass   1    28.086
pair_style   matpl  0_torch_script_module.pt 1_torch_script_module.pt
 ↪  2_torch_script_module.pt 3_torch_script_module.pt  out_freq
 ↪  ${DUMP_FREQ} out_file model_devi.out
pair_coeff       * * 14

variable        NSTEPS          equal 400
variable        THERMO_FREQ     equal 5
variable        TEMP            equal 500.000000
variable        PRESS           equal 100.000000
variable        TAU_T           equal 0.100000
variable        TAU_P           equal 0.500000

neighbor        1.0 bin
neigh_modify    delay 10
```

```
box               tilt large
change_box        all triclinic


thermo_style    custom step temp pe ke etotal press vol lx ly lz xy xz
 ↪  yz
thermo          ${THERMO_FREQ}

fix             1 all npt temp ${TEMP} ${TEMP} ${TAU_T} iso ${PRESS}
 ↪  ${PRESS} ${TAU_P}

timestep        0.001000
dump            1 all custom ${DUMP_FREQ} traj/*.lammpstrj id type x y
 ↪  z fx fy fz
restart         10000 lmps.restart

if "${restart} == 0" then "velocity        all create ${TEMP} 75740"
run             ${NSTEPS} upto
```

6.3.11.5.1 sys_idx 用于设置 md 的初始结构，为 list 格式，值为 sys_configs中的结构下标，这里可以指定多个结构。

6.3.11.5.2 select_sys 与 sys_idx 配合使用，用于限制 sys_idx 中每个初始探索结构的最多用于标注的构型数量，默认不设置，采用 max_select中的设置。如果参数 max_select 也未设置，将采用默认值 100。例如对于如下设置：

```
  "sys_idx": [0, 1],
  "select_sys":[20, 30],
```

sys_idx 指定了 0 号结构 POSCAR 和 1 号结构 49.config，在 0 号结构对应的轨迹中最多选取 20 个结构用于标注，在 1 号结构对应的轨迹中最多最多选取 30 个结构用于标注。如果不设置 select_sys，则对 0 和 1 号结构分别最多选取max_select个结构做标记。

您也可以设置为"select_sys":20，等效于"select_sys":[20, 20]。

6.3.11.5.3 trj_freq 用于设置轨迹采样频率（thermo），默认值为 10，即间隔 10 步采样一次。

6.3.11.5.4 lmps_in_idx 配合 lmps_in 使用。设置对sys_idx中对应的初始结构做分子动力学探索的 lammps.in 文件所在路径。使用如下例子中所示：

```
        "md_jobs": [
            [{
```

```
            "sys_idx": [ 1,3,4],
            "select_sys":[10,15,20],
            "lmps_in_idx":[0, 1, 2],
            "trj_freq": 5
        },{
            "sys_idx": [0, 1],
            "lmps_in_idx":3
        }
        ]]
```

在该例中，对于" sys_idx" : [1,3,4] 指定了需要探索的结构，即上例中的 49.config、41.config、37.config。分别设置 lammps 控制文件为 in0.lmps、in1.lmps、in2.lmps。设置每隔 5 步，采样一次。

对于" sys_idx" : [0, 1] 指定的结构 49.config、41.config，设置 lammps 输入控制文件 in3.lmps。采样间隔用默认值，每个 10 步采样一次。

存在 lmps_in_idx 设置时，`ensemble`、`nsteps`、`md_dt`、`press`、`taup`、`temps`、`taut` 参数将自动失效。

6.3.11.5.5 ensemble 用于第一种 lammps 输入控制方式。设置 系综，默认值 `"nve"`, 支持如下设置:

`npt`、`npt-i` 或 `npt-iso` 对应 lammps 设置

```
fix  1 all npt temp ${TEMP} ${TEMP} ${TAU_T} iso ${PRESS} ${PRESS}
 ↪  ${TAU_P}
```

`npt-a` 或 `npt-aniso` 对应 lammps 设置

```
fix  1 all npt temp ${TEMP} ${TEMP} ${TAU_T} aniso ${PRESS} ${PRESS}
 ↪  ${TAU_P}
```

`npt-t`、`npt-tri` 对应 lammps 设置

```
fix  1 all npt temp ${TEMP} ${TEMP} ${TAU_T} tri ${PRESS} ${PRESS}
 ↪  ${TAU_P}
```

`nvt` 对应 lammps 设置

```
fix  1 all nvt temp ${TEMP} ${TEMP} ${TAU_T}
```

`nve` 对应 lammps 设置

```
fix  1 all nve
```

6.3.11.5.6 nsteps 用于第一种 lammps 输入控制方式。设置 md 总的步数，为必选参数，需要用户提供。##### md_dt

用于第一种 lammps 输入控制方式。用于设置 `timestep` ，默认值为 `0.001`，即 `1` 飞秒。

6.3.11.5.7 press 用于第一种 lammps 输入控制方式。用于设置 md 探索的压强，为 list 格式。

6.3.11.5.8 taup 用于第一种 lammps 输入控制方式。稳压器的耦合时间（ps），默认值 `0.5`。

6.3.11.5.9 temps 用于第一种 lammps 输入控制方式。用于设置 md 探索的温度，为 list 格式。

6.3.11.5.10 taut 用于第一种 lammps 输入控制方式。恒温器的耦合时间（ps），默认值 `0.1`。

6.3.11.5.11 boundary 用于第一种 lammps 输入控制方式。设置模拟系统的边界条件，由于 MatPL 力场只支持对周期性体系的模拟，因此该值为 `true`，即采用 `p p p`。不需要用户设置。

6.3.11.5.12 例子

```
"explore": {
    "sys_config_prefix": "./init_bulk/collection/init_config_0",
    "sys_configs": [
        {"config":"0.95_scale.poscar", "format":"vasp/poscar"},
        {"config":"0_pertub.poscar", "format":"vasp/poscar"},
        {"config":"0_pertub.poscar", "format":"vasp/poscar"}
    ],
    "md_jobs": [
        [{
            "ensemble": "npt",
            "nsteps": 1000,
            "md_dt": 0.002,
            "trj_freq": 10,
            "sys_idx": [0, 1],
            "temps": [500, 800],
            "taut":0.1,
            "press" :[ 100,200],
            "taup": 0.5,
            "boundary":true
            },{
            "ensemble": "nvt",
            "nsteps": 1000,
            "md_dt": 0.002,
            "trj_freq": 10,
            "sys_idx": [2],
            "temps": [400],
            "taut":0.1,
            "boundary":true
```

```
        }]
    ]
}
```

在上例中，配置了一个轮次的主动学习，执行 "md_jobs" 中两个 dict 中配置的 lammps 模拟。

对于第一个 dict, 使用 npt 系综，在 sys_idx 配置了两个构型，对应 0.95_scale.poscar 和 0_pertub.poscar。温度和压强的列表分别为 [500, 800] 和 [100,200]，意思是对这两个结构分别在温度、压强组合为 [500, 100]、[500, 800]、[800, 100]、[800, 200] 下执行 lammps 模拟，模拟 1000 步，输出频率 10 步，单步时间长度 2 飞秒。模拟结束后，会获得 8 条轨迹。

对于第二个 dict，使用 nvt 系综，在 sys_idx 配置了 1 个构型，对应 0_pertub.poscar，温度为 400，即在 温度 400K 下做 lammps 模拟。模拟结束后获得 1 条轨迹。

下面是" ensemble" : "npt" , "nsteps" : 1000, "md_dt" : 0.002, "trj_freq" : 10, "sys_idx" : 0, "temps" : 500, "taut" :0.1, "press" :100, "taup" : 0.5, "boundary" :true 设置下，自动生成的 lammps.in 输入控制文件内容:

```
variable        NSTEPS          equal 400
variable        THERMO_FREQ     equal 5
variable        DUMP_FREQ       equal 5
variable        restart         equal 0
variable        TEMP            equal 500.000000
variable        PRESS           equal 100.000000
variable        TAU_T           equal 0.100000
variable        TAU_P           equal 0.500000

units           metal
boundary        p p p
atom_style      atomic

neighbor        1.0 bin
neigh_modify    delay 10

box              tilt large
if "${restart} > 0" then "read_restart lmps.restart.*" else
 ↪  "read_data lmp.config"
change_box       all triclinic

mass   1    28.086
pair_style   matpl  0_torch_script_module.pt 1_torch_script_module.pt
 ↪  2_torch_script_module.pt 3_torch_script_module.pt  out_freq
 ↪  ${DUMP_FREQ} out_file model_devi.out
pair_coeff       * * 14
```

```
thermo_style    custom step temp pe ke etotal press vol lx ly lz xy xz
 ↪  yz
thermo          ${THERMO_FREQ}
dump            1 all custom ${DUMP_FREQ} traj/*.lammpstrj id type x y
 ↪  z fx fy fz
restart         10000 lmps.restart

if "${restart} == 0" then "velocity        all create ${TEMP} 76752"
fix             1 all npt temp ${TEMP} ${TEMP} ${TAU_T} iso ${PRESS}
 ↪  ${PRESS} ${TAU_P}

timestep        0.001000
run             ${NSTEPS} upto
```

文件中 velocity 中的 76752 为随机生成值。### DFT

设置自洽计算，为 dict 格式。

6.3.11.6 dft_style 设置标注 (自洽计算) 使用哪种 DFT 计算软件，默认值为 pwmat，也支持 VASP 或 CP2K 格式，分别设置为 `vasp` 或 `cp2k`。如果设置为 `bigmodel`，则必须在 `bigmodel_script` 中设置处理脚本。

6.3.11.7 bigmodel_script 用于设置 大模型做标记（推理能量和受力）时的处理脚本。

对大模型标记的接口设置，请参考例子 si_direct_bigmodel。

6.3.11.8 input 设置输入控制文件的路径，可以为绝对路径或相对路径（相对于当前路径）。

6.3.11.9 kspacing 该参数为 PWMAT 的输入参数，用于设置 K 点，可选参数。如果在 etot.input 文件中未设置 `MP_N123` 参数，则使用该参数设置 K 点。不能同时设置 `MP_N123` 与 `kspacing`。

如果 `etot.input` 文件中未设置 `MP_N123`，且 `kspacing` 未设置，则采用默认设置 `kspacing` 值为 `0.5`。

6.3.11.10 flag_symm 该参数为 PWMAT 的输入参数，用于设置 K 点，可选参数。对于 Relax 或者 SCF 计算，默认值为 0，对于 AIMD 计算，默认值为 3。

6.3.11.11 pseudo 设置 PWMAT 或 VASP 赝势文件所在路径，为 list 格式，赝势文件路径可以为绝对路径或相对路径（相对于当前路径）。

6.3.11.12 gaussian_param CP2K 或 PWMAT 高斯基组参数设置，basis_set_file 和 potential_file 指定基组和势函数文件路径。atom_list, basis_set_list, potential_list 配合使用，分别指定元素对应的基组和势函数设置。kspacing 用于设置 K 点，用法与 PWMAT KSPACKING 设置 相同。

```
"gaussian_param": {
    "basis_set_file":"./init_bulk/BASIS_MOLOPT_1",
    "potential_file":"./init_bulk/POTENTIAL_1",
    "atom_list":["Si"],
    "kspacing" :0.4,
    "basis_set_list":["SZV-MOLOPT-SR-GTH"],
    "potential_list":["GTH-PBE-q4"]
}
```

6.3.11.13 DFT 设置例子 由于自洽计算任务使用的输入控制相同，因此只需要单文件的设置，对于不同的 DFT 软件，分别设置如下。

6.3.11.13.1 PWMat 设置 对于 PWMAT，设置与 init_bulk 中相似，如果您未在 scf_etot.input 中指定“MP_N123”参数，则您需要设置 kspacing 和 flag_symm 参数。

```
  "DFT": {
          "dft_style": "pwmat",
          "input": "scf_etot.input",
          "kspacing":0.5,
          "flag_symm":0,
          "pseudo" : ["../Ag.SG15.PBE.UPF","../Au.SG15.PBE.UPF"],
          "_flag":"1 个整数，or scf 0 , aimd 3, 磁性体系 2"
  }
```

6.3.11.13.2 VASP 设置

```
  "DFT": {
          "dft_style":"vasp",
          "input":"INCAR_scf",
          "pseudo" : ["../Ag_POTCAR", "../Au_POTCAR"]
  }
```

6.3.11.13.3 CP2K 设置

```
  "DFT": {
          "dft_style":"cp2k",
          "input":"scf.inp",
          "gaussian_param": {
```

```
            "basis_set_file":"../BASIS_MOLOPT_1",
            "potential_file":"../POTENTIAL_1",
            "atom_list":["Ag", "Au"],
            "basis_set_list":["SZV-MOLOPT-SR-GTH-q11",
             ↪  "SZV-MOLOPT-SR-GTH-q11"],
            "potential_list":["GTH-PBE", "GTH-PBE"],
            "_kspacing":0.5
        }
  }
```

注意，请在 scf_cp2k.inp 文件中设置 IGNORE_CONVERGENCE_FAILURE 参数，设置后对于不收敛的情况，CP2K 将正常退出，pwact 在检测到 CP2K 正常退出后，将会继续向下执行。否则，pwact 在检测到 CP2K 异常停止后，将重新提交任务三次，三次失败后，将会停止运行。

6.3.12 run param.json 案例参考

如下例子，为一个标准的主动学习流程，两个轮次的主动学习，采用多模型委员会查询策略。更多使用案例，请参考源码根目录的 example。

```
{
  "reserve_work": false,
  "reserve_md_traj": false,
  "reserve_scf_files": false,

  "init_data": ["/path/init_data"],

  "train": {
    "_train_input_file": "std_si.json",

    "model_type": "DP",
    "atom_type": [14],
    "seed": 2023,
    "recover_train": true,
    "model": {
      "descriptor": {
        "Rmax": 6.0,
        "Rmin": 0.5,
        "M2": 16,
        "network_size": [25, 25, 25]
      },
      "fitting_net": {
        "network_size": [50, 50, 50, 1]
      }
    },
```

```
    "optimizer": {
      "optimizer": "LKF",
      "epochs": 10,
      "batch_size": 16,
      "print_freq": 10,
      "block_size": 5120,
      "kalman_lambda": 0.98,
      "kalman_nue": 0.9987,
      "train_energy": true,
      "train_force": true,
      "train_virial": false,
      "pre_fac_force": 2.0,
      "pre_fac_etot": 1.0,
      "pre_fac_virial": 1.0
    }
  },

  "strategy": {
    "uncertainty": "committee",
    "lower_model_deiv_f": 0.05,
    "upper_model_deiv_f": 0.15,
    "model_num": 4,
    "max_select": 10
  },

  "explore": {
    "sys_config_prefix": "/path/structures",
    "sys_configs": [
      { "config": "POSCAR", "format": "vasp/poscar" },
      "atom1.config",
      "atom2.config",
      "atom3.config",
      "atom4.config"
    ],
    "md_jobs": [
      [
        {
          "ensemble": "nvt",
          "nsteps": 1000,
          "md_dt": 0.002,
          "trj_freq": 10,
          "taup": 0.5,
          "sys_idx": [0, 1],
```

```
          "temps": [500, 700],
          "taut": 0.1,
          "boundary": true
        },
        {
          "ensemble": "npt",
          "nsteps": 1000,
          "md_dt": 0.002,
          "trj_freq": 10,
          "press": [100.0, 200.0],
          "taup": 0.5,
          "sys_idx": [0, 3],
          "temps": [500, 700],
          "taut": 0.1,
          "boundary": true
        }
      ],
      {
        "ensemble": "nvt",
        "nsteps": 4000,
        "md_dt": 0.002,
        "trj_freq": 10,
        "press": [100.0, 200.0],
        "taup": 0.5,
        "sys_idx": [0, 1],
        "temps": [500, 700],
        "taut": 0.1,
        "boundary": true
      }
    ]
  },
  "DFT": {
    "dft_style": "pwmat",
    "input": "scf_etot.input",
    "kspacing": 0.5,
    "flag_symm": 0,
    "pseudo": ["path/Si.SG15.PBE.UPF"]
  }
}
```

6.4 resource 参数设置

6.4.1 resource.json 文件

设置计算集群资源，包括对训练、分子动力学（MD）、DFT 计算（SCF、Relax、AIMD）使用的计算节点、CPU、GPU 资源以及对应的运行软件（Lammps、VASP、PWMAT、MatPL）。

所有可设置参数按照用途分为 `train`, `explore`, DFT, `direct` 四种模块，每个模块中的参数意义相同。

对于初始训练集制备（init_bulk）可设置 `explore`, DFT, `direct`，这里以 mcloud 环境设置为例。explore 用于设置大模型 MD 环境，direct 用于设置 direct 采样环境，DFT 用于设置对结构做 SCF 或 AIMD 所用环境。

```
{
    "DFT": {
        "command":"mpirun -np 4 PWmat",
        "task_run_num":1,
        "number_node": 1,
        "cpu_per_node": 4,
        "gpu_per_node": 4,
        "group_size": 1,
        "queue_name": "3080ti,3090",
        "custom_flags": [
        ],
        "_custom_flags": [
        ],
        "module_list": [
            "compiler/2022.0.2",
            "mkl/2022.0.2",
            "mpi/2021.5.1",
            "cuda/11.6",
            "pwmat"
        ]
    },

    "explore": {
        "command": "python sevennet_md.py",
        "group_size": 1,
        "number_node": 1,
        "gpu_per_node": 1,
        "cpu_per_node": 1,
        "queue_name": "3080ti",
        "custom_flags": [],
        "source_list": [
```

```
            "/share/app/anaconda3/envs/SevenNet/env.sh"
        ],
        "module_list": [
        ],
        "env_list": [
        ]
    },

    "direct": {
        "command": "python direct.py",
        "group_size": 1,
        "number_node": 1,
        "gpu_per_node": 1,
        "cpu_per_node": 1,
        "queue_name": "3080ti",
        "custom_flags": [],
        "source_list": [
            "/share/app/anaconda3/envs/m3gnet/env.sh"
        ],
        "module_list": [
        ],
        "env_list": [
        ]
    }
}
```

对于主动学习（run）可设置 `train`, `explore`, `direct`, DFT 四个模块，这里以 mcloud 环境设置为例。train 用于设置训练环境，explore 用于设置 lammps 分子动力学环境，direct 用于设置 direct 采样环境，DFT 用于设置对结构做 SCF 环境 或者 大模型 MD 环境。做：

```
{
  "train": {
    "command": "MatPL",
    "group_size": 1,
    "number_node": 1,
    "gpu_per_node": 1,
    "cpu_per_node": 1,
    "queue_name": "3080ti,3090",
    "custom_flags": [
    ],
    "source_list": [
      "/share/app/MATPL/MatPL-2025.3/env.sh"
    ],
    "module_list": [
```

```
    ],
    "env_list":[
    ]
  },
  "explore": {
    "command": "mpirun -np 1 lmp_mpi -in in.lammps",
    "group_size": 1,
    "number_node": 1,
    "gpu_per_node": 1,
    "cpu_per_node": 1,
    "queue_name": "3080ti,3090",
    "custom_flags": [],
    "source_list": [],
    "module_list": [
      "lammps4matpl/2025.3"
    ],
    "env_list":[]
  },
  "DFT": {
    "command": "mpirun -np 4 PWmat",
    "number_node": 1,
    "cpu_per_node": 4,
    "gpu_per_node": 4,
    "group_size": 1,
    "queue_name": "3080ti,3090",
    "custom_flags": [],
    "source_list": [],
    "module_list": [],
    "env_list":[
      "compiler/2022.0.2",
      "mkl/2022.0.2",
      "mpi/2021.5.1",
      "cuda/11.6",
      "pwmat"
    ]
  }
}
```

6.4.2 参数细节

参数可以分为 3 类。用于设置运行命令的 command;

用于设置每个计算任务的资源数量 number_node、cpu_per_node、gpu_per_node、group_size、queue_name、custom_flags;

用于加载软件和环境变量 custom_flags、source_list、module_list、env_list。

详细解释如下。### command

必选参数，设置模块对应命令。

不同任务的设置，例子：

对于 DFT 计算设置

```
PWmat 设置:
"command":"mpirun -np 4 PWmat"
VASP 设置:
"command":"vasp_std"
cp2k 设置:
"command":"mpirun -np $SLURM_NTASKS cp2k.popt"
```

对于 Lammps 计算设置：

```
"command":"mpirun -np 10 lmp_mpi"
```

-np 后面为使用的 CPU 数量，需要与cpu_per_node 设置保持一致，如果 gpu_per_node设置，那么 np 中设置的 cpu 任务将平均分配到 gpu_per_node 所设置的 GPU 上。

对于 MatPL 文档 模型训练设置：

```
"command":"MatPL"
```

6.4.3 group_size

该参数用于多个计算任务提交时的分组，同组内的计算任务将顺序执行。组间任务并行。

例如，对于 34 个 自洽计算任务，"group_size":5，此时将把 34 个自洽计算任务分为 6 个组，即 6 个 slurm 任务，每个 slrum 任务包含 5 个自洽计算（最后一个组有 4 个计算任务）。执行时，6 个 slurm 任务将同时提交到计算集群（slurm 任务内部的 5 个自洽计算将串行执行）。

- 对于训练或探索任务，默认值为 1，每组 1 个任务，即所有任务同时提交。
- 对于 AIMD 或 SCF 任务，即 DFT 下的参数，默认值为-1，所有任务将分到一个组内，执行时，每个任务串行执行（即任务执行完毕后再提交执行下一个任务）。建议您根据实际的任务数量设置该参数，避免同时提交大量 DFT 任务，挤占所有计算资源。

6.4.4 number_node

用于设置每个 slurm 任务的计算节点数量，默认值为 1，即 1 个计算节点。

在 train 模块，该参数自动设置 1。

6.4.5 gpu_per_node

用于设置每个节点下使用的 GPU 数目，默认值为 0，如果使用 PWMAT 做 DFT 计算（自洽计算、驰豫或者 AIMD）该值需要与 "command" 中设置的 GPU 数量一致。

在 train 模块，该参数自动设置 1。

6.4.6 cpu_per_node

用于设置每个节点使用的 CPU 数目，默认值为 1，注意该值需要 “>= gpu_per_node”。

在 train 模块，该参数自动设置 1。

6.4.7 queue_name

必选参数，用于设置使用的计算机群分区，为 "," 分割的字符串列表，例如 "queue_name":"cpu, 3080ti, 3090"。

6.4.8 custom_flags

用于设置 Slurm 脚本中的其他 #SBATCH 参数，可选参数，为 list 格式。例如，对于

```
"custom_flags": [
  "#SBATCH -x gn43,gn66"
]
```

在执行，将会把 "#SBATCH -x gn43,gn66" 自动拼接到 Slrum 脚本中。这里 “#SBATCH” 也可以省略，只写 “-x gn43,gn66”。

例如对于设置：

```
"number_node": 1,
"cpu_per_node": 4,
"gpu_per_node": 4,
"queue_name": "3080ti,3090",
"custom_flags": [
  "#SBATCH -x gn43,gn66"
]
```

生成的 slurm 头文件为：

```
#SBATCH --nodes=1
#SBATCH --gres=gpu:4
#SBATCH --ntasks-per-node=4
#SBATCH --partition=3080ti,3090
#SBATCH -x gn43,gn66
```

由于部分机器不支持--gpus-per-task 参数设置，因此 pwact 不会自动生成该行配置，如果需要设置，请写到 custom_flags 中即可。

6.4.9 source_list

用于设置 slurm 脚本在运行时需要导入的环境变量，可选参数，list 格式。例如，对于

```
"source_list": [
    "~/anaconda3/etc/profile.d/conda.sh"
]
```

在执行时，把字符串自动拼接前缀“source”后，将”source /opt/rh/devtoolset-8/enable”自动写入 slrum 脚本中。

6.4.10 module_list

用于设置 slurm 脚本在运行时需要加载的软件模块，可选参数，list 格式。

例如，对于

```
"module_list": [
    "cuda/11.8",
    "intel/2020"
]
```

在执行时，将会把字符串“module load cuda/11.8”和“module load intel/2020”自动写入 slurm 脚本中。

6.4.11 env_list

用于设置 slurm 脚本在运行时需要加载的环境信息，可选参数，list 格式。

例如，对于

```
"env_list": [
    "source the/path/MatPL-2025.3/env.sh"
]
```

在执行时，会将这条字符串完整写入 slurm 脚本中。

按照上述 queue_name、custom_flags、source_list、module_list、env_list 中的设置，生成的 slurm 脚本内容如下：

```
#SBATCH --partition=3080ti,3090
#SBATCH -x gn43,gn66
```

```
source /opt/rh/devtoolset-8/enable
module load cuda/11.8
module load intel/2020
source the/path/MatPL-2025.3/env.sh
```

6.4.12 配置案例详解-train 模块

对于 train 模块，需要加载 MatPL 的 Python 运行环境，如果使用 MCLOUD 上已安装的 MatPL 做训练，对应的设置如下：

```
"train": {
  "command": "MatPL",
  "group_size": 1,
  "number_node": 1,
  "gpu_per_node": 1,
  "cpu_per_node": 1,
  "queue_name": "3080ti,3090",
  "custom_flags": [
  ],
  "source_list": [
  "/share/app/MATPL/MatPL-2025.3/env.sh"
  ],
  "env_list": [
  ],
  "module_list": [
  ]
}
```

这里对于每个训练任务使用 1 个计算节点，使用该节点的 1 张 GPU,1 个 CPU，该节点位于分区 3080ti 或 3090。

如果从 MatPL 源码编译安装，对应的设置如下：

```
"train":{
  "command": "MatPL",
  "group_size": 1,
  "number_node": 1,
  "gpu_per_node": 1,
  "cpu_per_node": 1,
  "queue_name": "3080ti,3090",
  "custom_flags": [
  ],
  "source_list": [
      "~/anaconda3/etc/profile.d/conda.sh",
```

```
        "the/path/MatPL-2025.3/env.sh"
    ],
    "env_list": [
        "conda activate matpl-2025.3"
    ],
    "module_list": [
        "cuda/11.8-share",
        "intel/2020"
    ]
  }
```

这里 "~/anaconda3/etc/profile.d/conda.sh" 为笔者计算集群中的 conda 加载路径，matpl-2025.3 为 MatPL 的 Python 环境，the/path/MatPL-2025.3 为源码所在路径。

6.4.13 配置案例详解-explore 模块

对于 explore 模块，如果使用 MCLOUD 已安装的 LAMMPS 为例，直接加载 lammps4matpl 软件即可，完整的设置如下：

```
    "explore": {
      "command": "mpirun -np 1 lmp_mpi",
      "group_size": 2,
      "number_node": 1,
      "gpu_per_node": 1,
      "cpu_per_node": 1,
      "queue_name": "3080ti,3090",
      "custom_flags": [
      ],
      "source_list": [

      ],
      "module_list": [
          "lammps4matpl/2025.3"
      ],
      "env_list":[

      ]
    }
```

这里对于每个 lammps 任务，使用 1 个计算节点，使用该节点的 1 张 GPU,1 个 CPU，每 2 个 lammps 任务分为 1 个组。

如果从 LAMMPS 源码编译安装，对应的设置如下：

```
  "explore": {
```

```
    "command": "mpirun -np 1 lmp_mpi",
    "group_size": 2,
    "number_node": 1,
    "gpu_per_node": 1,
    "cpu_per_node": 1,
    "queue_name": "3080ti,3090",
    "custom_flags": [
    ],
    "source_list": [
        "the/path/of/lammps/env.sh"
    ],
    "module_list": [
        "cuda/11.8-share",
        "intel/2020"
    ],
    "env_list":[
    ]
    }
```

这里 `the/path/of/lammps/env.sh` 为 lammps 源码所在路径，lammps 力场接口

6.4.14 配置案例详解-DFT 模块

对于 DFT 模块，这里以加载 PWMAT 为例，设置如下。

```
  "DFT": {
      "command":"PWmat",
      "number_node": 1,
      "cpu_per_node": 4,
      "gpu_per_node": 4,
      "group_size": 5,
      "queue_name": "3080ti,1080ti,3090",
      "custom_flags": [
      "#SBATCH -x gn18,gn17"
      ],
      "module_list": [
          "compiler/2022.0.2",
          "mkl/2022.0.2",
          "mpi/2021.5.1",
          "cuda/11.6"
      ],
      "env_list":[
      ]
  }
```

这里对于每个 PWmat 任务，使用 1 个计算节点，使用该节点的 1 张 GPU,1 个 CPU，每 5 个 DFT 任务分为 1 个组。

6.5 硅体系的主动学习案例

本案例为硅体系的主动学习过程，案例位于 pwact/example/si_pwmat/。对于 Mcloud 用户，请访问路径/share/public/PWMLFF_test_data/pwact_examples/25-pwact-demo 即可。

案例首先通过 init_bulk 构造初始训练集，之后使用初始训练集训练模型，并使用在 init_bulk 中使用微扰产生的结构做为初始构型在 300K 、500K 和 900K 做主动学习采样。

请注意，案例中提供的 DFT 设置仅用于程序执行流程测试，不保证计算精度。

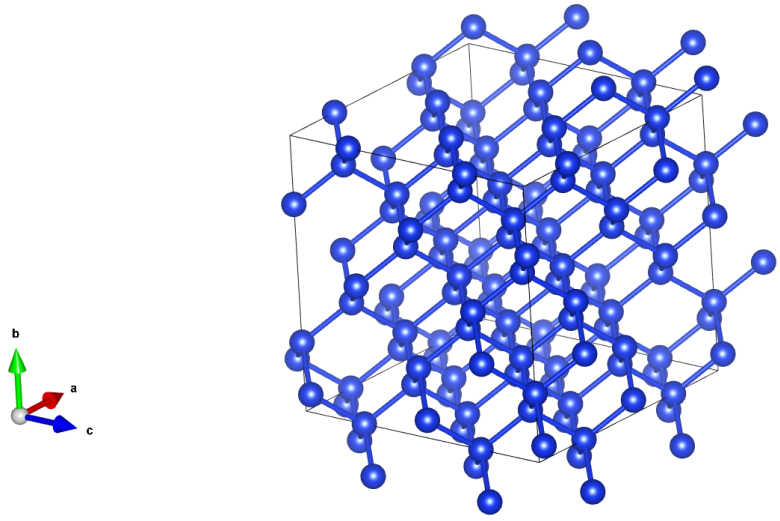


Figure 15: 硅体相

6.5.1 init_bulk 案例

启动命令如下

进入 pwact/example/si_pwmat/init_bulk 目录

```
pwact init_bulk init_param.json resource.json
```

6.5.2 init_bulk 目录结构

init_bulk 的目录结构如下所示，atom.config、POSCAR、resource.json、init_param.json、relax_etot.input、relax_etot1.input、aimd_etot1.input、aimd_etot2.input 为输入文件，collection 为执行后的结果汇总目录。

6.5.3 init_bulk 目录结构- collection 目录

datapath.txt 文件内容是预训练数据所在目录记录。

init_config_0 目录为 atom.config 进过驰豫、阔胞、缩放、微扰、aimd 后的结果汇总。

relax.config 是对 atom.config 做驰豫后得到的结构文件；super_cell.config 是对 relax.config 做 阔胞后得到结构文件；0.9_scale.config 和 0.95_scale.config 是对 super_cell.config 做晶格缩放后得到的结构文件；

0.9_scale_pertub 目录包括对 0.9_scale.config 结构做晶格和原子位置微扰后得到的 30 个结构；

train.xyz 目录是对微扰后的结构做按照 aimd_etot1.input 做 AIMD 之后，将得到轨迹提取成 extxyz 格式后的结果目录。如果是提取为 pwmlff/npy 格式，则为 PWdata 目录。内容如下所示。atom_type.npy 是结构的原子类型，position.npy 是结构中原子的位置信息，energies.npy、forces.npy、ei.npy、virials.npy 为结构的总能量、原子三个方向的力、原子能量和维里信息。ei.npy、virials.npy 是可选文件，如果轨迹中不包括原子能和维里，则不提取。

```
./init_config_0/PWdata/Si128
├── atom_type.npy
├── energies.npy
├── ei.npy
├── forces.npy
├── image_type.npy
├── lattice.npy
├── position.npy
└── virials.npy
```

6.5.4 init_bulk 目录结构- init_bulk 目录结构图

```
example/init_bulk
├──atom.config
├──POSCAR
├──resource.json
├──init_param.json
├──relax_etot.input
├──relax_etot1.input
├──relax_etot2.input
├──aimd_etot1.input
├──aimd_etot2.input
└──collection
    ├──datapath.txt
```

```
├──init_config_0
│    ├──super_cell.config
│    ├──0.9_scale.config
│    ├──0.9_scale_pertub
│    │        ├──0.9_scale.config
│    │        ├──0_pertub.config
│    │        ├──1_pertub.config
│    │        ├──2_pertub.config
│    │        ...
│    │        └──30_pertub.config
│    ├──0.95_scale.config
│    ├──0.95_scale_pertub
│    │        ├──0.95_scale.config
│    │        ├──0_pertub.config
│    │        ├──1_pertub.config
│    │        ├──2_pertub.config
│    │        ...
│    │        └──30_pertub.config
│    └──train.xyz
├──init_config_1
...
```

6.5.5 run 主动学习案例

我们使用 init_bulk 案例中的预训练数据和微扰后的结构，在 500K、800K 和 1100K 下做主动学习。

启动命令: 执行完毕 init_bulk 命令之后，进入 pwact/example/si_pwmat/ 目录:

```
pwact run param.json resource.json
```

6.5.6 run 主动学习文件目录

主动学习的目录结构如下所示。

param.json 和 resource.json 为主动学习输入控制文件，scf_etot.input 为主动学习做自洽计算的输入文件。

si.al 为主动学习流程记录文件，记录已执行的主动学习流程。

iter_result.txt 为主动学习每个轮次中探索和选取结构用于标注的记录。内容如下例所示。第一行为本轮次的选点情况。Total structures 为采集到的结构数量 = ccurate 准确点 + selected 候选点数量 + error 错误点数量。第二行为 MD 轨迹的总结，以例子中所示，表示共运行了 16 条 MD 轨迹，其中 16 条 MD 正确运行结束，0 条轨迹运行失败。运行失败的轨迹汇总在 error_traj.log 文件中。运行失败的情况一

般是力场不够准确，导致 MD 运行崩溃。接下来的行为标记过程中，已收敛和未收敛的结构数量统计以及未收敛的结构路径，对于为收敛的结构，在标记结束后，不会加入到训练集中。

```
iter.0000  Total structures 1296    accurate 6 rate 0.46%    selected
 ↪  23 rate 1.77%    error 1267 rate 97.76%
A total of 16 MD trajectories were run. with 16 trajectories
 ↪  correctly executed and 0 trajectories normally completed.
For detailed information, refer to File error_traj.log.

Number of converged files: 194, number of non-converged files: 3
List of non-converged files:
/share/public/PWMLFF_test_data/pwact_examples/25-pwact-
 ↪  demo/auag_vasp/run_iter_lmps/iter.0000/temp_run_iter_work/02.label/scf/md.00
 ↪  scf/OUTCAR
...

iter.0001  Total structures 12816    accurate 598 rate 4.67%
 ↪  selected 2926 rate 22.83%    error 9292 rate 72.50%
A total of 16 MD trajectories were run. with 16 trajectories
 ↪  correctly executed and 0 trajectories normally completed.
For detailed information, refer to File error_traj.log.

Number of converged files: 194, number of non-converged files: 3
List of non-converged files:
/share/public/PWMLFF_test_data/pwact_examples/25-pwact-
 ↪  demo/auag_vasp/run_iter_lmps/iter.0001/temp_run_iter_work/02.label/scf/md.00
 ↪  scf/OUTCAR
...
```

`iter.0000` 为第一轮次的主动学习目录，`iter.0001` 为第二轮次的主动学习目录，以此类推。

`00.train`、`01.explore`、`02.label` 为主动学习轮次中对应的模型训练、探索、标记三个任务所在目录。

6.5.7 run 00.train 目录

对于 `00.train` 目录，这里采用 4 模型的委员会查询策略，训练 4 个模型，这 4 个模型只有网络参数的初始化值不同，其他完全相同。`0-train.job`、`1-train.job`、`2-train.job`、`3-train.job` 为执行训练任务的 4 个 slurm 任务脚本。训练任务执行完毕后，将生成 4 个标识任务执行成功的 tag 文件 (`0-tag.train.success`、`1-tag.train.success`、`2-tag.train.success`、`3-tag.train.success`)，以及 4 个模型，目录为 `train.000`、`train.001`、`train.002`、`train.003`。

以 `train.000` 目录为例，`train.json` 为 MatPL 模型训练的输入文件。`std_input.json` 为 MatPL 输出的训练参数设置汇总。`model_record` 为模型的保存目录，`dp_model.ckpt` 为

模型文件，epoch_train.dat 为在每个 epoch 下的平均训练误差，为每个 epoch 训练结束后在验证集上的平均误差，保存在 epoch_valid.dat。

torch_script_module.pt 为训练结束后，使用 jitscript 工具编译 dp_model.ckpt 后的模型文件，做为力场，用于接下来在 lammps 中的模拟。

6.5.8 run 01.explore 目录

01.explore 目录包括两个子目录，md 和 select 子目录。

md 目录为主动学习的探索目录，包括在不同温度、压强等条件下对初始结构调用 MatPL 力场做分子动力学模拟的文件（输入文件、轨迹）。

之后把模拟得到的结构（轨迹）根据委员会查询方法的上下界设置做筛选，筛选的结果保存在 select 子目录。

6.5.9 run 01.explore 目录- md 子目录

md 子目录包括两个子目录，目录名称为 md.***.sys.***/md.***.sys.***.t.***，例如 md.000.sys.000/md.000.sys.000.t.000，对于 md.000.sys.000 目录，这里 md.000 的 000 指 param.json 中的md_jobs 对第 0 个 md 设置；sys.000 为sys_index 的下标 0 对应的结构。包括 md.000.sys.000.t.000、md.000.sys.000.t.001 两个子目录，分别表示在温度 temps 对应下标为 0 和 1 温度下的分子动力学模拟。

在每个 md.*.sys.* 目录下都有一个 model_devi_distribution.png 文件，是对该目录下所有轨迹的偏差值分布统计绘图。

在最后一级子目录下，是对应 MD 设置以及轨迹文件：

```
0_torch_script_module.pt, 1_torch_script_module.pt,
 ↪  2_torch_script_module.pt, 3_torch_script_module.pt,
 ↪  atom_type.txt, in.lammps, lmp.config, log.lammps, md.log,
 ↪  model_devi.out, tag.md.success
```

model_devi.out 格式：

```
#    step        avg_devi_f       min_devi_f       max_devi_f
 ↪  avg_devi_e       min_devi_e       max_devi_e
        0      0.008682422      0.000574555      0.014187865
 ↪  0.030333196      0.011702489      0.043806718
       10      0.016611307      0.000627126      0.027185410
 ↪  0.031661788      0.005340899      0.048241921
       20      0.028689107      0.000811486      0.040269587
 ↪  0.036761117      0.000214928      0.059791462
       30      0.046457606      0.001416745      0.063635973
 ↪  0.049553956      0.000165127      0.081012839
```

```
      40       0.065051169       0.000681319       0.089395358
↪  0.065790218       0.000036582       0.103522832
```

model_devi.out 的第 1 列为当前步编号，第 2 列为力偏差的均值，第 3 列为力偏差最小值，第 4 列为力偏差最大值。力偏差最大值计算公式如下所示：

$$\varepsilon_t = \max_i \sqrt{\left\langle \sum_{w=1}^{W} \| F_{w,i}(R_t) - \hat{F}_i \|^2 \right\rangle}, \quad \hat{F}_i = \frac{1}{W} \sum_{w=1}^{W} F_{w,i}$$

这里 W 为模型数量，i 为原子下标，$<>$ 为取平均。

后三列分别是原子能量的偏差均值、最小值和最大值。在 PWact 主动学习中使用的是第 4 列的最大力偏差值。

6.5.10 run 01.explore 目录- select 子目录

以下.csv 文件内容包含 3 列，分别是力偏差（devi_force）、结构在轨迹中对应的编号（config_index）、轨迹的文件路径（file_path）。

accurate.csv 是力偏差小于设置的力偏差下限的结构汇总。

fail.csv 是力偏差大于设置的力偏差上限的结构。

如果候选的结构（力偏差介于设置的力偏差上下限之间的结构）超过设置的最大选点数量 max_select，则将候选的结构随机选取 max_select 个，存入 candidate.csv 文件，其余的存入 candidate_delete.csv 文件。

model_devi_distribution-md.*.sys.*.png 为超链接文件，是探索结构的偏差值分布绘图。

error_traj.log 是 MD 过程为执行 MD 过程中断的轨迹，中断的原因一般是力场不准确造成原子丢失、原子靠的太近等。

select_summary.txt 是筛选点的数据量信息汇总，内容如下例所示。

```
Total structures 862   accurate 0 rate 0.00%   selected 616 rate 71.46%   err

Select by model deviation force:
Accurate configurations: 0, details in file accurate.csv
Candidate configurations: 616, randomly select 27, delete 589
        Select details in file candidate.csv
        Delete details in file candidate_delete.csv.
Error configurations: 246, details in file fail.csv

A total of 10 MD trajectories were run. with 8 trajectories correctly execute
For detailed information, refer to File error_traj.log.
```

6.5.11 run 02.label 目录

02.label 目录包括 scf 和 result 两个子目录。scf 为对 explore 中筛选出的点做自洽计算的目录。自洽计算后，将结构和对应的能量、力、原子能、维里提取为 pwdata 格式，保存在 result 子目录，作为后面主动学习轮次的训练数据。

6.5.12 run 02.label 目录- scf

scf 下的一级和二级子目录 md.*.sys.*/md.*.sys.*.t.* 与 md 子目录的结构和名称意义完全相同。在二级子目录下,是自洽计算的目录。以 scf/md.000.sys.000/md.000.sys.000.t.000/820-scf 目录为例，这里 820 指 md.000.sys.000/md.000.sys.000.t.000 轨迹的第 820 步对应的结构。该目录下，820.config 为自洽计算的输入结构，etot.input 为自洽计算的输入控制文件，REOPORT 和 OUT.MLMD 为 PWMAT 的输出文件。其中 OUT.MLMD 文件包括了自洽计算后的结构原子位置信息、能量、力等信息。

6.5.13 run 02.label 目录- result

result 为标记结束后的带标签数据集汇总，如果设置 data_format 为 extxyz 格式，将提取为 train.xyz 文件。如果为 pwmlff/npy 格式，则为 PWdata 目录，目录下是具体数据。

6.5.14 run 主动学习目录结构图

```
example
├──param.json
├──resource.json
├──scf_etot.input
├──si.al
├──iter_result.txt
├──iter.0000
│     ├──00.train
│     │    ├──0-train.job
│     │    ├──1-train.job
│     │    ├──3-train.job
│     │    ├──2-train.job
│     │    ├──train.000
│     │    │     ├──train.json
│     │    │     ├──std_input.json
│     │    │     ├──model_record
│     │    │     │     ├──dp_model.ckpt
│     │    │     │     ├──epoch_train.dat
```

```
│   │   │   │   └──epoch_valid.dat
│   │   │   ├──torch_script_module.pt
│   │   │   └──tag.train.success
│   │   ├──train.001
│   │   │   └──...
│   │   ├──train.002
│   │   │   └──...
│   │   ├──train.003
│   │   │   └──...
│   │   ├──0-tag.train.success
│   │   ├──1-tag.train.success
│   │   ├──2-tag.train.success
│   │   └──3-tag.train.success
│   ├──01.explore
│   │   ├──md
│   │   │   ├──md.000.sys.000
│   │   │   │   ├──md.000.sys.000.t.000.p.000
│   │   │   │   ├──...
│   │   │   │   └──model_devi_distribution.png
│   │   │   ├──md.000.sys.001
│   │   │   ├──md.001.sys.000
│   │   │   └──md.001.sys.003
│   │   └──select
│   │       ├──accurate.csv
│   │       ├──candidate.csv
│   │       ├──candidate_delete.csv
│   │       ├──fail.csv
│   │       ├──error_traj.log
│   │       ├──select_summary.txt
│   │       ├──model_devi_distribution-md.000.sys.000.png
│   │       └──...
│   └──02.label
│       ├──scf
│       │   ├──md.000.sys.000
│       │   │   ├──md.000.sys.000.t.001
│       │   │   │   ├──820-scf
│       │   │   │   │   ├──820.config
│       │   │   │   │   ├──etot.input
│       │   │   │   │   ├──REPORT
│       │   │   │   │   └──OUT.MLMD
```

```
│   │   │   │   │   ├──200-scf
│   │   │   │   │   └──...
│   │   │   │   └──md.000.sys.000.t.000
│   │   │   │       └──...
│   │   │   ├──md.000.sys.001
│   │   │   ├──...
│   │   │   ├──md.001.sys.000
│   │   └──result
│   │       └──train.xyz

├──iter.0001
│   └──...
├──...
```

6.6 硅体系案例（大模型采样）

本案例为硅体系的主动学习过程，采用了大模型（eqv2）和 direct 采样。在 PWact/examples/ 下准备了多种组合设置，每种组合的信息请参考PWact/examples/README.md。对于 Mcloud 用户，请访问路径/share/public/PWMLFF_test_data/pwact_examples/25-pwact-demo 即可。这里以 [PWact/examples/si_pwmatgaussion_bigmodel_direct] 为例介绍大模型和 direct 方法的参数设置、脚本设置。对于 init_bulk 构造初始训练集，以examples/si_pwmatgaussion_bigmodel_direct/init_bulk_bigmodel为例：

- step1. 用 PWmat (gaussion 基组) 做 relax;
- step2. 对结构调用大模型（seventnet）做分子动力学;
- step3. 对分子动力学得到的轨迹做 direct 采样，去掉轨迹中相似的结构，筛选出的结构用于后续主动学习

请注意，本目录中使用的是 gaussian 基组的 PWmat，仅用于快速测试，例子生成的数据不具有可靠性。

6.6.1 预训练数据制备 init_bulk

启动命令: 进入 examples/si_pwmatgaussion_bigmodel_direct/init_bulk_bigmodel 目录，这里提供了在 mcloud 上的执行脚本，也可以通过如下命令执行。

```
pwact init_bulk init_param.json resource.json
```

6.6.2 init_bulk 目录结构

init_bulk 目录与 si_pwmat 案例 相似，只是多了一项 bigmodel 目录，如下所示。

```
.
├── datapath.txt
├── init_config_0/
├── init_config_1/
└── bigmodel/
    ├── 0-bigmodel.job
    ├── 0-tag.bigmodel.success
    ├── 1-bigmodel.job
    ├── 1-tag.bigmodel.success
    ├── ...
    ├── direct/
    │   ├── 0-direct.job
    │   ├── 0-tag.direct.success
    │   ├── candidate.json
    │   ├── candidate.xyz
    │   ├── Cov_score.png
    │   ├── direct.py
    │   ├── PCA_direct.png
    │   ├── PCA_variance.png
    │   ├── PWdata
    │   ├── select_idx.dat
    │   ├── select.xyz
    │   └── tag.direct.success
    ├── init_config_0/
    │   ├── 0.95_scale/
    │   │   ├── 3_bigmodel/
    │   │   │   ├── npt.log
    │   │   │   ├── POSCAR
    │   │   │   ├── sevennet_md.py
    │   │   │   ├── tag.bigmodel.success
    │   │   │   ├── tmp.traj
    │   │   │   └── traj.xyz
    │   │   ├── 4_bigmodel/
    │   │   └── 5_bigmodel/
    │   ├── 0.9_scale/
    │   └── 1.0_scale/
    └── init_config_*/
```

- `*-bigmodel.job` 和 `*-tag.bigmodel.success` 为 大模型运行 MD 的 slurm 脚本和执行成功的 tag 标记文件
- `init_config_0`、`init_config_1`、...目录为执行大模型 MD 的工作目录

- direct 目录为执行 direct 方法筛选结构的工作目录

6.6.3 init_bulk 接入 大模型 MD

对于大模型 MD，提供了用户自定义接口，要求用户配置大模型的运行环境，可参考例子中使用的sevennet_md.py。

如目录 bigmodel/init_config_0/0.95_scale/3_bigmodel/ 所示，在 init_bulk 运行过程中,pwact 检查到存在 bigmodel 设置时,会把对应结构按照类似 bigmodel/init_config_0/0.95_sca 录所示生成，包括一个 VASP/POSCAR 格式的 POSCAR 文件，拷贝用户提供的运行 MD 脚本的接口文件到该目录，如 sevennet_md.py 文件，根据用户在 resource.json 中设置的 explore，自动生成 slurm 脚本，提交运行。

用户需要在脚本中读取 POSCAR 文件后运行 MD，将生成的轨迹转换为 extxyz 格式的文件，文件名称固定为 traj.xyz。之后，pwact 将检测该文件，用于后续处理。

6.6.4 init_bulk 接入 direct 采样

对于 direct 采样，提供了用户自定义接口，要求用户 direct 方法的运行环境，可参考例子中使用的direct.py。

如目录 bigmodel/direct/ 所示，在 大模型 MD 运行结束后，会生成多条轨迹文件，pwact 会自动探测轨迹文件，将文件合并为 一个 名为 candidate.xyz 的文件，放到目录 bigmodel/direct 下。拷贝用户在 direct_input 提供的运行 direct 脚本的接口文件到该目录，如 direct.py 文件，根据用户在 resource.json 中设置的 direct，自动生成 slurm 脚本，提交运行。

用户需要在脚本中读取 candidate.xyz 文件，之后需要生成 select_idx.dat 和 select.xyz 两个文件，其中 select_idx.dat 保存筛选出的结构在 candidate.xyz 中对应的下标，select.xyz 保存筛选出的结构。pwact 会自动读取这两个文件用于后续的执行。

6.6.5 主动学习 run

我们使用 init_bulk 案例中的预训练数据和微扰后的结构，在 500K、800K 和 1100K 下做主动学习。

启动命令:执行完毕 init_bulk 命令之后,进入 examples/si_pwmatgaussion_bigmodel_direct/run_ 目录:

```
pwact run param.json resource.json
```

6.6.6 主动学习文件目录

主动学习的目录结构与 si_pwmat 例子中相似。

6.6.7 train 目录

train 目录 与 si_pwmat 例子 相同

6.6.8 explore 目录

explore 目录除了 md 和 select 子目录外，新增加了一个名为 bigmodel 的子目录，子目录内容与 init_bulk direct 中相同。

md 和 select 子目录与 si_pwmat 例子 相同

6.6.9 label 目录

如果设置了使用大模型做标注bigmodel_script，则在 label 目录下新增一个 bigmodel 子目录。该子目录结构如下所示:

```
bigmodel/
├── 0-bigmodel.job
├── eqv2_label.py
├── select.xyz
└── train.xyz
```

- eqv2_label.py 为 用户设置的 bigmodel_script 脚本文件，pwact 运行时会将改文件拷贝到 bigmodel 目录下
- select.xyz 为 大模型标注的输入结构文件，格式为 extxyz
- train.xyz 为 大模型标注后的输出结构文件，格式为 extxyz，相比于 select.xyz，结构顺序相同，但是多了 能量和力的信息
- 0-bigmodel.job 为 pwact 根据用户在 resource.json 中的 DFT 设置，生成的 slurm 脚本

6.6.10 run 接入 大模型 标记

对于 大模型 标记，提供了用户自定义接口，要求用户提供 大模型 的运行环境，可参考例子中使用的eqv2_label.py。

如目录 bigmodel/ 所示，在 explore 步骤，经过多模型偏差筛选 (或继续使用 direct 筛选) 后，pwact 会自动探测筛选出的结构，将文件合并为 一个 名为 select.xyz 的文件，放到目录 bigmodel/ 下。拷贝用户在 bigmodel_script 提供的运行 bigmodel 脚本的接口文件到该目录，如 eqv2_label.py 文件，根据用户在 resource.json 中设置的 DFT，自动生成 slurm 脚本，提交运行。

用户需要在脚本中读取 select.xyz 文件，之后需要生成 train.xyz 文件，train.xyz 相比于 select.xyz 多了能量和力信息。pwact 会自动读取这该文件用于后续的执行。

6.6.11 result

`result` 为标记结束后的带标签数据集汇总，如果设置 `data_format` 为 `extxyz` 格式，将提取为 train.xyz 文件。如果为 `pwmlff/npy` 格式，则为 PWdata 目录，目录下是具体数据。

6.7 金银合金体系案例

本案例为金银合金体系的主动学习过程，案例位于 `pwact/example/auag_pwmat/` 首先通过 `init_bulk` 构造初始训练集，之后使用初始训练集训练模型，使用在 init_bulk 中使用微扰产生的结构做为初始构型，使用用户输入的 `lammp.in` 输入控制文件做主动学习采样。:::tip 使用用户输入的 `lammp.in` 输入控制文件下做主动学习采样在 `>= pwact-0.4` 版本中开始支持。:::

以下案例使用的 DFT 计算软件为 PWMAT,案例也提供了使用 VASP 的对应设置`pwact/example/auag_vasp/`、CP2K 的对应设置`pwact/example/auag_cp2k/`。对于 Mcloud 用户，请访问路径`/share/public/PWMLFF_test_data/pwact_examples/25-pwact-demo` 即可。

请注意，案例中提供的 DFT 设置仅用于程序执行流程测试，不保证计算精度!

6.7.1 预训练数据制备 init_bulk

启动命令: 进入 `pwact/example/auag_pwmat/init_bulk` 目录

```
pwact init_bulk init_param.json resource.json
```

6.7.2 init_bulk 目录结构

init_bulk 目录与 si_pwmat 案例 目录结构相同。

6.7.3 主动学习 run

启动命令: 执行完毕 init_bulk 命令之后，进入 `examples/auag_pwmat/run_iter_lmps` 目录

```
pwact run param.json resource.json
```

6.7.4 主动学习文件目录

主动学习目录结构与 si_pwmat 例子目录结构相同。唯一区别是对 md 的目录名称，如'md.000.sys.000/md.000.sys.000.t.000.p.000'将变成'md.000.sys.000/md.000.sys.000.lmps.000'，这里 `lmps.000` 用于标示使用的 lammps.in 文件编号。

```
example
├──param.json
├──resource.json
├──scf_etot.input
├──iter.0000
│     ├──00.train
│     │    ├──...
...
│     ├──01.explore
│     │    ├──md
│     │    │     ├──md.000.sys.000
│     │    │     │     ├──md.000.sys.000.lmps.000
│     │    │     │     ├──...
│     │    │     │     └──model_devi_distribution.png
│     │    │     ├──md.000.sys.001
│     │    │     ├──md.001.sys.000
│     │    │     └──md.001.sys.003
│     │    └──select
│     │          ├──accurate.csv
│     │          ├──candidate.csv
│     │          ├──candidate_delete.csv
│     │          ├──fail.csv
│     │          ├──error_traj.log
│     │          ├──select_summary.txt
│     │          ├──model_devi_distribution-md.000.sys.000.png
│     │          └──...
│     └──02.label
│     │    ├──scf
│     │    │     ├──md.000.sys.000
│     │    │     │     ├──md.000.sys.000.t.001
│     │    │     │     │     ├──820-scf
│     │    │     │     │     │     ├──820.config
│     │    │     │     │     │     ├──etot.input
│     │    │     │     │     │     ├──REPORT
│     │    │     │     │     │     └──OUT.MLMD
│     │    │     │     │     ├──200-scf
│     │    │     │     │     └──...
│     │    │     │     └──md.000.sys.000.t.000
│     │    │     │           └──...
│     │    │     ├──md.000.sys.001
│     │    │     ├──...
│     │    │     ├──md.001.sys.000
│     │    └──result
│     │          └──train.xyz
```

```
├──iter.0001
│     └──...
├──...
```

7 pwdata 数据转换工具

7.1 pwdata 数据转换工具介绍

当前手册页使用 `pwdata >= 0.5.0`

pwdata 是 MatPL 的数据预处理工具，包括如下功能：

- `atom.config(PWmat)`、`POSCAR(VASP)`、`lmp.init(Lammps)`、`cp2k.init(CP2K)` 之间的文件互转；
- 对这些结构做阔胞、晶格缩放、晶格或原子位置微扰；
- 提取各种轨迹文件 `MOVEMENT(PWmat)`、`OUTCAR(VASP)`、`lammps dump file(Lammps)`、`cp2k md file(CP2K)` 或常用训练数据 `pwmlff/npy`、`extxyz`、`deepmd/npy`、`deepmd/raw`、`meta` OMAT24 开源数据集之后转换为 `pwmlff/npy` 或者 `extxyz` 格式文件。对于 meta 数据集，增加了 cpu 并行和查询操作，以快速从超过一亿结构的数据库中查找自己想要的结构。

7.1.1 支持的数据类型

Software	file	multi-Image	label	format
PWmat	MOVEMENT	True	True	‘pwmat/movement’
PWmat	OUT.MLMD	False	True	‘pwmat/movement’
PWmat	atom.config	False	False	‘pwmat/config’
VASP	OUTCAR	True	True	‘vasp/outcar’
VASP	poscar	False	False	‘vasp/poscar’
LAMMPS	lmp.init	False	False	‘lammps/lmp’
LAMMPS	dump	True	False	‘lammps/dump’
CP2K	stdout, xyz, pdb	True	True	‘cp2k/md’
CP2K	stdout	False	True	‘cp2k/scf’
MatPL	*.npy	True	True	‘pwmlff/npy’
DeepMD (read)	*.npy, *.raw	True	True	‘deepmd/npy’ , ‘deepmd/raw’
* (extended xyz)	*.xyz	True	True	‘extxyz’
Meta (read)	*aselmdb	True	True	‘meta’

7.1.2 安装方式

pip 命令安装:

```
pip install pwdata

# 安装 pwdata, 如果已安装，则升级到最新版本
pip install pwdata --upgrade

# 列出所有可安装版本
pip index versions pwdata
# 输出结果示例:
# pwdata (0.3.2)
# Available versions: 0.3.2, 0.3.1, 0.3.0, 0.2.16, 0.2.15
#   INSTALLED: 0.3.2
#   LATEST:    0.3.2
# 安装指定版本
pip install pwdata==n.m.o
```

7.2 pwdata 命令行调用方式

pwdata 支持 命令行操作 以及 源码接入 两种方式。

命令列表：pwdata 命令列表如下，您可以通过‘-h’选项获取该命令列表的详细解释以及所有支持参数。这里 pwdata -h 用于输出所有可用命令，以及该命令的使用例子。

```
pwdata -h
pwdata convert_config or cvt_config -h
pwdata convert_configs or cvt_configs -h
pwdata scale_cell or scale -h
pwdata super_cell or super -h
pwdata perturb -h
pwdata count -h
```

7.2.1 pwdata convert_config 结构互转

该命令用于各种结构之间的互转，您可以使用 convert_config 或者它的缩写 cvt_config 参数如下所示

```
pwdata convert_config [-h] -i INPUT -f INPUT_FORMAT -o OUTPUT_FORMAT
 ↪  [-s SAVENAME] [-c] [-t ATOM_TYPES [ATOM_TYPES ...]]
```

7.2.1.1 -h 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.1.2 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径

7.2.1.3 -f 可选参数，输入文件的格式，如不指定将根据输入文件自动推测输入格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp' , 'cp2k/scf']

7.2.1.4 -o 必选参数，输出文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp']

7.2.1.5 -s 输出文件的名称，与 -o 配合使用。如果未指定，对于 pwmat/config 格式将使用 atom.config 作为文件名，对于 vasp/poscar 格式使用 POSCAR, 对于 lammps/lmp 格式使用 lammps.lmp 作为文件名

7.2.1.6 -c 是否使用分数坐标，如果指定了 -c 参数，在保存结构时将使用笛卡尔坐标，否则使用分数坐标。注意，pwmat/config 只支持分数坐标，此时该参数将失效

7.2.1.7 -t 输入结构的原子类型，该参数用于当输入结构是 lammps/lmp 或 lammps/dump 格式时指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致。对于其他格式的输入文件，该参数失效

我们在源码的根目录下提供了 examples，您可以下载后使用这些测试例子：

```
# convert_config 案例: atom.config 转 poscar
# 执行完毕后将在 examples/test_workdir 目录下生成结构文件 cvtcnf_atom.POSCAR
pwdata cvt_config -i examples/pwmat_data/LiGePS_atom.config -s
 ↪  examples/test_workdir/cvtcnf_atom.POSCAR -o vasp/poscar
```

7.2.2 pwdata convert_configs 训练数据转换

提取各种轨迹文件 MOVEMENT(PWmat)、OUTCAR(VASP)、lammps dump file(Lammps)、cp2k md file(CP2K) 或常用训练数据 pwmlff/npy、extxyz、deepmd/npy、deepmd/raw、meta OMAT24 开源数据集为 pwmlff/npy 或者 extxyz 格式文件。您可以使用 convert_configs 或者它的缩写 cvt_configs.

参数如下所示

```
pwdata [-h] -i INPUT [INPUT ...] [-f INPUT_FORMAT] [-s SAVEPATH] [-o
 ↪  OUTPUT_FORMAT] [-r]
              [-m MERGE] [-g GAP] [-q QUERY] [-n CPU_NUMS] [-t
               ↪  ATOM_TYPES [ATOM_TYPES ...]]
```

7.2.2.1 -h 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.2.2 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径。该参数为列表形式，支持输入多个文件路径或者目录。pwdata 对 pwmlff/npy、extxyz、deepmd/npy、deepmd/raw、meta OMAT24 开源数据集实现了目录自动查询，您只需要指定数据源根目录即可。

例如对于examples/pwmlff_data/LiSiC，该目录下存在’ C2, C448, C448Li75, C64Si32, Li1Si24, Li3Si8, Li8, Li88Si20, Si1, Si217’ 这些子目录, 输入'-i examples/pwmlff_data' 即可。

此外，如果您的文件或者目录较多（如 meta 数据库这类非常多的子文件），您也可以将这些路径写入一个 json 文件中，如下所示，命令中指定-i jsonfile 即可。

```
{
    "datapath" : [
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-1000-
        ↪  subsampled",
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-300",
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-300-
        ↪  subsampled"
    ]
}
```

7.2.2.3 -f 可选参数，输入文件的格式，pwdata 实现了对输入数据的格式自动推理，不需要显式指定输入格式。支持的格式有 [‘pwmat/movement’ ， ‘vasp/outcar’ ， ‘lammps/dump’ ， ‘cp2k/md’ ， ‘pwmlff/npy’ ， ‘deepmd/npy’ ， ‘deepmd/raw’ ， ‘extxyz’ ， ‘meta’]

7.2.2.4 -o 可选参数，输出文件的格式，支持的格式有 [‘pwmlff/npy’ ，‘extxyz’]，默认格式为'pwmlff/npy'

7.2.2.5 -s 可选参数，输出文件的目录，如不指定，将使用当前目录

7.2.2.6 -m 可选参数，int 类型，该参数仅用于输出文件格式为 extxyz，设置 '-m 1' 所有结构会保存到一个 xyz 文件中，默认保存到一个 xyz 文件中。设置 '-m 0' 将按照元素类型保存到不同的 xyz 文件中，

7.2.2.7 -g 可选参数，int 类型，该参数用于提取轨迹文件时，指定每隔多少步取一帧结构。默认值为 1

7.2.2.8 -q 可选参数，str 类型，该参数值在输入类型为 meta 时生效，用于查询数据库操作，对于 -q 指定的参数，建议用请用引号包裹，避免特殊字符如 H>4，shell 会把 > 符号解释为重定向符号，详细的使用参考 meta 查询演示。

7.2.2.9 -n 可选参数，int 类型，该参数在输入类型为 meta 时生效，用于设置并行查询数据库时使用的 CPU 核数，默认使用单核

7.2.2.10 -t 如果输入格式是 lammps/lmp 或 lammps/dump 格式，该参数用于指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致

如果输入格式是 meta，该该参数用于查找所有只存在这些元素类型的结构。

7.2.3 pwdata convert_configs 案例

我们在源码的根目录下提供了 examples，您可以下载后使用这些测试例子：

例 1. 将目录 examples/pwmlff_data/LiSiC 下的所有 pwmlff/npy 格式数据提取为 xyz 格式，其中训练集占 80%，测试集占 20%，执行完毕后，在 examples/test_workdir/0_1_configs_extxyz 目录下会产生 train.xyz 和 valid.xyz 两个文件。

```
pwdata convert_configs -i examples/pwmlff_data/LiSiC -s
 ↪  examples/test_workdir/0_1_configs_extxyz -o extxyz
```

例 2. 将 PWmat 轨迹文件 50_LiGePS_movement 和 lisi_50_movement 每隔 5 步提取一帧，随机划分 80% 结构做为训练集 20% 作为测试集，存在 examples/test_workdir/3_1_configs_extxyz 目录下

```
pwdata convert_configs -i examples/pwmat_data/50_LiGePS_movement
 ↪  examples/pwmat_data/lisi_50_movement -s
 ↪  examples/test_workdir/3_1_configs_extxyz -o extxyz -g 5
```

例 3. 将 examples/deepmd_data/alloy 目录下的所有 deepmd/npy 格式文件提取为 pwmlff/npy 格式，不划分测试集

```
pwdata convert_configs -i examples/deepmd_data/alloy -s
 ↪  ./test_workdir/7_0_configs_PWdata
```

例 4. 将 examples/xyz_data 目录下的所有后缀为 xyz 的文件提取为 pwmlff/npy 格式，随机划分 80% 和 20% 作为训练和测试集，保存在 examples/test_workdir/5_0_configs_PWdata 目录

```
pwdata convert_configs -i examples/xyz_data -s
 ↪  examples/test_workdir/5_0_configs_PWdata -g 1
```

例 5. 在 examples/meta_data/alex_val 目录下所有后缀为.aselmdb 的 meta 数据库中，查询元素类型为 Pt 和 Ge 的结构, 将查询到的所有结构保存到./test_workdir/10_1_configs_extxyz 目 录下不划分测试集

```
pwdata convert_configs -i examples/meta_data/alex_val -s
 ↪  ./test_workdir/10_1_configs_extxyz -t Pt Ge
```

例 6. 在 meta_data['data_path'] 中列出的所有目录或者文件下的所有后缀为.aselmdb 的 meta 数据库中，查询元素类型为 Pt 和 Ge 的结构，把查询到的所有结构保存到./test_workdir/ 10_1_configs_extxyz 目录下，不划分测试集

```
pwdata convert_configs -i examples/meta_data.json -s
 ↪  ./test_workdir/10_1_configs_extxyz -o extxyz -t Pt Ge
```

7.2.4 pwdata convert_configs meta 查询例子

例 1. 查询只包含 Pt 和 Ge 两种元素的结构，并将查询结果输出到为 xyz 格式。执行完成后将会在 examples/test_workdir/10_1_configs_extxyz 目录下生成一个 train.xyz 和 valid.xyz 两个文件

```
pwdata convert_configs -i
 ↪  examples/meta_data/alex_val/alex_go_aao_001.aselmdb
 ↪  examples/meta_data/alex_val/alex_go_aao_002.aselmdb -s
 ↪  examples/test_workdir/10_1_configs_extxyz -o extxyz -t Pt Ge
```

例 2. 使用-q 参数查询，查询包含了 Cu 元素的所有结构

```
pwdata convert_configs -i
 ↪  examples/meta_data/alex_val/alex_go_aao_001.aselmdb
 ↪  examples/meta_data/alex_val/alex_go_aao_002.aselmdb -s
 ↪  examples/test_workdir/10_1_configs_extxyz -o extxyz -q 'Cu'
```

例 3. 使用-q 参数查询，查询结构中 H 原子数目少于 3 个的所有结构

```
pwdata convert_configs -i
 ↪  examples/meta_data/alex_val/alex_go_aao_001.aselmdb
 ↪  examples/meta_data/alex_val/alex_go_aao_002.aselmdb -s
 ↪  examples/test_workdir/10_1_configs_extxyz -o extxyz -q 'H<3'
```

例 4. 使用-q 参数查询，查询结构中，包含 Cu 原子并且 H 原子数目少于 3 个的所有结构

```
pwdata convert_configs -i
 ↪  examples/meta_data/alex_val/alex_go_aao_001.aselmdb
 ↪  examples/meta_data/alex_val/alex_go_aao_002.aselmdb -s
 ↪  examples/test_workdir/10_1_configs_extxyz -o extxyz -q 'Cu,H<3'
```

例 5. 使用-q 参数查询，查询结构中，至少包含 2 个 H 原子且至少包含 1 个 O 原子的所有结构

```
pwdata convert_configs -i
 ↪  examples/meta_data/alex_val/alex_go_aao_001.aselmdb
 ↪  examples/meta_data/alex_val/alex_go_aao_002.aselmdb -s
 ↪  examples/test_workdir/10_1_configs_extxyz -o extxyz -q 'H2O'
```

一些其他查询语句

```
| ---query string---| -------------------------Function
 ↪  Explanation---------------------------- |
| v3                | has  'v3'  key
 ↪  |
| abc=bla-bla       | has key  'abc'  with value  'bla-bla'
 ↪  |
| v3,abc=bla-bla    | both of the above
 ↪  |
| calculator=nwchem | calculations done with NWChem
 ↪  |
| 2.2<bandgap<3.0   |  'bandgap'  key has value between 2.2 and 3.0
 ↪  |
| natoms>=10        | 10 or more atoms
 ↪  |
| id=2345           | specific id
 ↪  |
| age<1h            | not older than 1 hour
 ↪  |
| age>1y            | older than 1 year
 ↪  |
| pbc=TTT           | Periodic boundary conditions along all three
 ↪  axes                          |
| pbc=TTF           | Periodic boundary conditions along the first two
 ↪  axes (F=False, T=True)    |
https://databases.fysik.dtu.dk/ase/ase/db/db.html#id7
```

pwdata 使用了输入格式的自动推测，以及数据目录自动查询，指定 `pwmlff/npy`、`extxyz`、`deepmd/npy`、`deepmd/raw`、`meta` `OMAT24` 开源数据集 这些数据源的根目录即可自动读取。

注意：pwdata 在自动查找数据源时，如您未显式指定数据源格式，pwdata 会将输入目录下的所有可用数据源都作为输入数据。

7.2.5 pwdata scale_cell 晶格缩放

该命令用于对结构的晶格做缩放，您可以使用 `scale_cell` 或者它的缩写 `scale` 参数如下所示

```
pwdata scale_cell [-h] -r SCALE_FACTOR [SCALE_FACTOR ...] -i INPUT -f
 ↪  INPUT_FORMAT [-s SAVENAME] [-o OUTPUT_FORMAT] [-c] [-t ATOM_TYPES
 ↪  [ATOM_TYPES ...]]
```

7.2.5.1 `-h` 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.5.2 -r 必选参数，晶格的缩放因子，$Lattic_{new} = factor * Lattice_{old}$，factor 取值范围 (0.0, 1.0)，浮点类型的列表，用空格隔开。例如 `'-r 0.97 0.98 0.99'` 表示对输入结构分别做 0.97、0.98、0.99 的晶格缩放，必选参数

7.2.5.3 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径，必选参数

7.2.5.4 -f 可选参数，如不显式指定，将根据输入文件自动推测。输入文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp' , 'cp2k/scf']，必选参数

7.2.5.5 -o 可选参数，输出文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp'], 如果未指定该参数, 将会使用输入结构的格式, 此时如果输入的文件格式是 `cp2k/scf`, 那么将使用 `pwmat/config` 格式

7.2.5.6 -s 可选参数，输出文件的名称，与 -o 配合使用，并将缩放因子作为前缀。如果未指定，对于 `pwmat/config` 格式将使用 `atom.config` 作为文件名，对于 `vasp/poscar` 格式使用 POSCAR , 对于 `lammps/lmp` 格式使用 `lammps.lmp` 作为文件名。例如 `'-o pwmat/config -s atom.config -r 0.99'` 缩放后的新文件名称为 `'0.99_atom.config'`

7.2.5.7 -c 可选参数，是否使用分数坐标，如果指定了 -c 参数，在保存结构时将使用笛卡尔坐标，否则使用分数坐标。注意，PWmat 只支持分数坐标，此时该参数将失效

7.2.5.8 -t 输入结构的原子类型，该参数用于当输入结构是 `lammps/lmp` 或 `lammps/dump` 格式时指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致。对于其他格式的输入文件，该参数失效

我们在源码的根目录下提供了 examples，您可以下载后使用这些测试例子：

```
# scale_cell 命令例子
pwdata scale_cell -r 0.98 0.99 0.97 0.95 -i
 ↪  examples/pwmat_data/lisi_atom.config -f pwmat/config -s
 ↪  examples/test_workdir/scale_atom.config -o pwmat/config

# 或命令的缩写
pwdata scale -r 0.98 0.99 0.97 0.95 -i
 ↪  examples/pwmat_data/lisi_atom.config -f pwmat/config -s
 ↪  examples/test_workdir/scale_atom.config -o pwmat/config

# 执行完毕后将在 examples/test_workdir 目录下生成 4 个缩放后的文件，分被名为
 ↪  0.98_scale_atom.config、0.99_scale_atom.config、
 ↪  0.97_scale_atom.config、0.95_scale_atom.config
```

7.2.6 pwdata super_cell 阔胞

该命令用于对结构的晶格做缩放，您可以使用 super_cell 或者它的缩写 super。参数如下所示

```
pwdata super_cell [-h] -m SUPERCELL_MATRIX [SUPERCELL_MATRIX ...] -i
 ↪  INPUT -f INPUT_FORMAT [-s SAVENAME] [-o OUTPUT_FORMAT] [-c] [-p
 ↪  PERIODICITY [PERIODICITY ...]] [-l TOLERANCE] [-t ATOM_TYPES
 ↪  [ATOM_TYPES ...]]
```

7.2.6.1 -h 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.6.2 -m 必选参数，超晶胞矩阵 (3x3)，3 个或者 9 个值，例如 '-m 2 0 0 0 2 0 0 0 2' 或者 '-m 2 2 2' 表示超晶胞是 2x2x2 的，必选参数

7.2.6.3 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径，必选参数

7.2.6.4 -f 可选参数，如不显式指定，将根据输入文件自动推测。输入文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp' , 'cp2k/scf']，必选参数

7.2.6.5 -o 可选参数，输出文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp'], 如果未指定该参数, 将会使用输入结构的格式, 此时如果输入的文件格式是 cp2k/scf, 那么将使用 pwmat/config 格式

7.2.6.6 -s 可选参数, 输出文件的名称, 与 -o 配合使用。如果未指定, 对于 pwmat/config 格式将使用 atom.config 作为文件名, 对于 vasp/poscar 格式使用 POSCAR , 对于 lammps/lmp 格式使用 lammps.lmp 作为文件名。例如 '-o pwmat/config -s super_atom.config' 缩放后的新文件名称为 'super_atom.config'

7.2.6.7 -c 可选参数，是否使用分数坐标，如果指定了 -c 参数，在保存结构时将使用笛卡尔坐标，否则使用分数坐标。注意，PWmat 只支持分数坐标，此时该参数将失效

7.2.6.8 -t 输入结构的原子类型，该参数用于当输入结构是 lammps/lmp 或 lammps/dump 格式时指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致。对于其他格式的输入文件，该参数失效

7.2.6.9 -p 可选参数，周期性边界条件。例如，[1, 1, 1] 表示系统在 x, y, z 方向上是周期性的。默认为 [1,1,1]

7.2.6.10 -l 可选参数，分数坐标的容差。默认为 1e-5。防止轻微负坐标被映射到模拟盒中

我们在源码的根目录下提供了 examples，您可以下载后使用这些测试例子：

```
# super_cell 命令例子
pwdata super_cell -m 2 3 4 -i examples/pwmat_data/lisi_atom.config -s
 ↪  examples/test_workdir/super_atom.config -o pwmat/config

# 或命令的缩写
pwdata super -m 2 0 0 0 3 0 0 0 4 -i
 ↪  examples/pwmat_data/lisi_atom.config -s
 ↪  examples/test_workdir/super_atom.config -o pwmat/config

# 执行完成后将在 examples/test_workdir 目录下生成一个名为 super_atom.config 的文
 ↪  件，采用了 2X3X4 阔胞
```

7.2.7 pwdata perturb 晶格和原子位置微扰

该命令用于对结构的晶格或者原子位置坐微扰。参数如下所示

```
pwdata perturb [-h] [-d ATOM_PERT_DISTANCE] [-e CELL_PERT_FRACTION]
 ↪  -n PERT_NUM -i INPUT -f  INPUT_FORMAT [-s SAVENAME] [-o
 ↪  OUTPUT_FORMAT] [-c] [-t ATOM_TYPES [ATOM_TYPES ...]]
```

7.2.7.1 -h 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.7.2 -d 可选参数，原子微扰的距离，决定原子相对原始位置的移动距离。对每个原子的三个坐标值，分别加上从 [-atom_pert_distance，atom_pert_distance] 范围内的均匀分布中随机采样的值。微扰是以埃为单位的距离。例如，0.01 表示原子的移动距离是 0.01 埃，默认值为 0，即不对原子位置微扰

7.2.7.3 -e 可选参数，晶胞变形的程度。对 9 个晶格值分别加上从 [-cell_pert_fraction，cell_pert_fraction] 范围内的均匀分布中随机采样的值。例如，0.03 表示晶胞变形的程度是相对原始晶胞的 3%，默认值为 0，即不对晶格做微扰

7.2.7.4 -n 必选参数，需要微扰的结构数量

7.2.7.5 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径，必选参数

7.2.7.6 -f 可选参数，如不显式指定，将根据输入文件自动推测。输入文件的格式，支持的格式有 [‘pwmat/config’，‘vasp/poscar’，‘lammps/lmp’，‘cp2k/scf’]，必选参数

7.2.7.7 -o 可选参数，输出文件的格式，支持的格式有 ['pwmat/config' , 'vasp/poscar' , 'lammps/lmp'], 如果未指定该参数, 将会使用输入结构的格式, 此时如果输入的文件格式是 cp2k/scf, 那么将使用 pwmat/config 格式

7.2.7.8 -s 可选参数, 输出文件的名称, 与 -o 配合使用。如果未指定, 对于 pwmat/config 格式将使用 atom.config 作为文件名, 对于 vasp/poscar 格式使用 POSCAR , 对于 lammps/lmp 格式使用 lammps.lmp 作为文件名。例如 '-o pwmat/config -s super_atom.config' 缩放后的新文件名称为 'super_atom.config'

7.2.7.9 -c 可选参数，是否使用分数坐标，如果指定了 -c 参数，在保存结构时将使用笛卡尔坐标，否则使用分数坐标。注意，PWmat 只支持分数坐标，此时该参数将失效

7.2.7.10 -t 输入结构的原子类型，该参数用于当输入结构是 lammps/lmp 或 lammps/dump 格式时指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致。对于其他格式的输入文件，该参数失效

我们在源码的根目录下提供了 examples，您可以下载后使用这些测试例子:

```
# perturb 命令例子
pwdata perturb -e 0.01 -d 0.04 -n 20 -i
 ↪  examples/pwmat_data/lisi_atom.config -f pwmat/config -s
 ↪  examples/test_workdir/perturb_atom -o pwmat/config
# 微扰后将在 examples/test_workdir/pertub_atom 目录下生成 20 个微扰后的结构
```

7.2.8 pwdata count 统计结构数量

统计各种轨迹文件 MOVEMENT(PWmat)、OUTCAR(VASP)、lammps dump file(Lammps)、cp2k md file(CP2K) 或常用训练数据 pwmlff/npy、extxyz、deepmd/npy、deepmd/raw、meta OMAT24 开源数据集为 pwmlff/npy 或者 extxyz 中的结构数量。

参数如下所示

```
pwdata convert_configs [-h] [-h] -i INPUT [INPUT ...] [-f
 ↪  INPUT_FORMAT] [-q QUERY] [-n CPU_NUMS] [-t ATOM_TYPES [ATOM_TYPES
 ↪  ...]]
```

7.2.8.1 -h 输出帮助信息，将列出命令的所有可用参数及其解释

7.2.8.2 -i 必选参数，输入文件的文件路径，支持绝对路径或者相对路径。该参数为列表形式，支持输入多个文件路径或者目录。pwdata 对 pwmlff/npy、extxyz、deepmd/npy、deepmd/raw、meta OMAT24 开源数据集实现了目录自动查询，您只需要指定数据源根目录即可。

例如对于examples/pwmlff_data/LiSiC，该目录下存在’ C2, C448, C448Li75, C64Si32, Li1Si24, Li3Si8, Li8, Li88Si20, Si1, Si217’ 这些子目录, 输入`'-i examples/pwmlff_data'` 即可。

此外，如果您的文件或者目录较多（如 meta 数据库这类非常多的子文件），您也可以将这些路径写入一个 json 文件中，如下所示，命令中指定`-i jsonfile` 即可。

```
{
    "datapath" : [
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-1000-
        ↪  subsampled",
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-300",
        "/share/public/PWMLFF_test_data/eqv2-
        ↪  models/datasets/decompress/Omat24/train/rattled-300-
        ↪  subsampled"
    ]
}
```

7.2.8.3 `-f` 可选参数，输入文件的格式，pwdata 实现了对输入数据的格式自动推理，不需要显式指定输入格式。支持的格式有 [‘pwmat/movement’ ， ‘vasp/outcar’ ， ‘lammps/dump’ ， ‘cp2k/md’ ， ‘pwmlff/npy’ ， ‘deepmd/npy’ ， ‘deepmd/raw’ ， ‘extxyz’ ， ‘meta’]

7.2.8.4 `-q` 可选参数，str 类型，该参数值在输入类型为 `meta` 时生效，用于查询数据库操作，详细的使用参考 meta 查询演示

7.2.8.5 `-n` 可选参数，int 类型，该参数在输入类型为 `meta` 时生效，用于设置并行查询数据库时使用的 CPU 核数，默认使用单核

7.2.8.6 `-t` 如果输入格式是 `lammps/lmp` 或 `lammps/dump` 格式，该参数用于指定结构的原子类型，可以是元素名称或者原子编号，顺序需要与输入结构中保持一致

如果输入格式是 `meta`，该该参数用于查找所有只存在这些元素类型的结构。

7.3 pwdata 作为独立工具使用 (接口调用)

pwdata 也可以作为一个独立的工具使用，通过调用 pwdata 的接口来生成数据集或者进行数据转换。

7.3.1 pwdata Config 类 从输入文件中读取数据

```
Config (self, format: str, data_path: str, pbc = None, atom_names =
↪  None, index = ':', **kwargs)
```

参数: - **format**: 字符串. 输入文件的格式。支持的格式有: `pwmat/config`, `vasp/poscar`, `lammps/dump`, `lammps/lmp`, `pwmat/movement`, `vasp/outcar`, `cp2k/md`, `cp2k/scf`, `pwmlff/npy`, `deepmd/npy`, `deepmd/raw`, `extxyz`、`meta`。

- `pwmat/config`: PWmat 结构文件，例如 `atom.config`.
- `pwmat/movement`: PWmat 分子动力学轨迹文件，例如 MOVEMENT.
- `lammps/dump`: LAMMPS dump 文件, 例如 `dump.lammptraj`.
- `lammps/lmp`: LAMMPS 结构文件，例如 `in.lmp`.
- `vasp/poscar`: VASP 结构文件，例如 POSCAR.
- `vasp/outcar`: VASP 分子动力学轨迹文件，例如 OUTCAR.
- `cp2k/md`: CP2K 标准输出文件，原子位置文件及对应的原子力文件，例如 `cp2k.out`, `*pos-1.xyz`, `*pos-1.pdb`, `*frac-1.xyz`.
- `cp2k/scf`: CP2K 标准输出文件，例如 `cp2k.out`.
- `pwmlff/npy`: MatPL 数据集文件，例如 `energies.npy`.
- `deepmd/npy`: DeepMD 数据集文件，例如 `force.npy`.
- `deepmd/raw`: DeepMD 数据集文件，例如 `force.raw`.
- `extxyz`: 扩展的 xyz 文件，例如 `*.xyz`.
- `meta`: meta 开源的数据文件，后缀名为`.aselmdb`.

CP2K 的输入控制文件中需要设置 `PRINT_LEVEL MEDIUM`，标准输出文件从才会存在晶格信息。

- **data_path**: 字符串, **必选**. The path of the input file.
- **pbc**: 列表, 可选. 周期性边界条件。默认为 None。例如，`[1, 1, 1]` 表示系统在 x, y, z 方向上是周期性的。
- **atom_names**: 列表, 可选. 用于 `lammps/dump` 文件的原子名称。默认为 None。例如，`['C', 'H', 'O']` 表示系统包含碳、氢和氧原子。
- **index**: 整数, 切片 或 字符串, 可选. 用于读取包含多个结构的文件时，可以通过 index 参数指定读取的结构。默认为 `:`，表示读取所有结构。
  - `index=0`: 第一个结构
  - `index=-2`: 倒数第二个结构
  - `index=':'` 或 `index=slice(None)`: 所有结构
  - `index='-3:'` 或 `index=slice(-3, None)`: 倒数第三个到最后一个结构
  - `index='::2'` 或 `index=slice(0, None, 2)`: 偶数个数的结构
  - `index='1::2'` 或 `index=slice(1, None, 2)`: 奇数个数的结构
- **kwargs**: 键值对（字典）, 可选. 其他关键字参数用于读取输入文件。
- **unit**: 字符串, 可选. 对于 LAMMPS 文件，输入文件的单位。默认为 ‘metal’。

- **style**: 字符串, 可选. 对于 LAMMPS 文件，用于 lammps 模拟种原子相关联的每个原子的属性。默认为 'atomic'。详情见 LAMMPS atom_style。
- **sort_by_id**: 布尔值, 可选. 对于 LAMMPS 文件，是否按照 id 排序原子。默认为 True。

返回: 返回一个 Image 对象的列表。每个 Image 对象包含一个结构的一些信息。

例子:

```
from pwdata import Config

data_file = "./cp2k.out"
format = "cp2k/scf"
config = Config(format, data_file)
```

7.3.2 pwdata Config.to() 数据写入文件

根据读入的文件格式，将数据保存为指定格式的文件

```
Config.to (self, data_path, format = None, **kwargs)
```

参数:

- **data_path**: 字符串, **必选**. 保存文件的路径。
- **format**: 字符串, **必选**. 保存文件的格式。默认为 None。支持的格式有 `pwmat/config`, `pwmat/movement`, `vasp/poscar`, `lammps/lmp`, `extxyz`, `pwmlff/npy`.
- **Kwargs**: 是其他用于保存文件的关键字参数。
- 用于以下格式的文件: `pwmat/config`, `vasp/poscar`, `lammps/lmp`, `extxyz`。
  - **data_name**: 字符串, **必选**. 结构文件的保存名称。
  - **sort**: bool, **可选**. 是否按照原子序数排序。默认为 None。
  - **wrap**: bool, 可选. 是否将原子映射到模拟盒中。默认为 False。
  - **direct**: bool, **必选**. 原子坐标是分数坐标还是笛卡尔坐标。(0 0 0) -> (1 1 1)
- 用于保存标签文件的关键字参数。用于 `pwmlff/npy` 格式的文件。
  - **data_name**: 字符串, **必选**. 数据集文件夹的保存名称。
  - **random**: bool, 可选. 是否对原始数据进行随机打乱，然后将数据划分为训练集和验证集。默认为 True。
  - **seed**: int, 可选. 随机数种子。默认为 2024。
- 输入为 CP2K 的数据时，`sort` 参数需要设置为 `False`，因为 CP2K 的数据已经是按照原子序数排序的，再次排序会导致数据顺序错误。
- `pwmlff/npy` 用于保存数据集的标签。它可以用于训练机器学习模型。
- PWmat 结构文件只能保存为分数坐标，不能保存为笛卡尔坐标，因此 `direct` 参数无效。

- LAMMPS 结构文件只能保存为笛卡尔坐标，不能保存为分数坐标，因此 `direct` 参数无效。

例子 1:

```
from pwdata import Config

data_file = "./POSCAR"
format = "vasp/poscar"
config = Config(format, data_file)
config.to(data_path = "./", format = "lammps/lmp", data_name =
 ↪  "lmp.init", direct = False, sort = True)
```

对于具有相同结构的多个配置，如果有需要的话，可以在调用 `.to()` 方法之前调用 `.append()` 方法将它们拼接在一起。

例如:

```
from pwdata import Config

raw_data = ["./OUTCAR0", "./OUTCAR1", "./OUTCAR2"]    # the same
 ↪  atoms...
format = "vasp/outcar"
multi_data = Config(format, raw_data[0])
for data in raw_data[1:]:
   image_data = Config(format, data)
   multi_data.append(image_data)
multi_data.to(data_path = "./PWdata", format='pwmlff/npy')
```

例子 2:

将‘pwmat/movement’、‘vasp/outcar’、‘cp2k/md’、或者‘lammps/dump’轨迹文件转换为单结构文件‘pwmat/config’、‘vasp/poscar’、‘lammps/lmp’

```
from pwdata import Config
from pwdata.utils.constant import FORMAT

def trajs2config():
    atom_types = ["Hf", "O"] # for lammps
    input_file =
 ↪  "/data/home/wuxingxing/codespace/pwdata/examples/lmps_data/HfO2/30.lammpstrj
    input_format="lammps/dump"
    save_format = "pwmat/config"
    image = Config(data_path=input_file, format=input_format,
 ↪  atom_names=atom_types)
    tmp_image_data = image.images
    save_dir = "./tmp_test"
    for id, config in enumerate(tmp_image_data):
```

```
        savename = "{}_{}".format(id,
↪   FORMAT.get_filename_by_format(save_format))
        image.iamges = [config]
        image.to(data_path = save_dir,
            data_name = savename,
            format = save_format,
            sort = True)

if __name__=="__main__":
    trajs2config()
```

7.3.3 pwdata build.supercells 阔胞

```
build.supercells.make_supercell (image_data, supercell_matrix: list,
↪   pbc: list = None, wrap=True, tol=1e-5)
```

根据输入的原始结构和超晶胞矩阵构建超晶胞。

参数:

- **image_data**: **必选**. Image 对象，包含原始结构的一些信息。
- **supercell_matrix**: 列表, **必选**. 超晶胞矩阵 (3x3)。例如，`[[2, 0, 0], [0, 2, 0], [0, 0, 2]]` 表示超晶胞是 2x2x2 的。
- **pbc**: 列表, 可选. 周期性边界条件。默认为 None。例如，`[1, 1, 1]` 表示系统在 x, y, z 方向上是周期性的。
- **wrap**: 布尔值, 可选. 是否将原子映射到模拟盒中（对于周期性边界条件）。默认为 True。
- **tol**: 浮点数, 可选. 分数坐标的容差。默认为 1e-5。防止轻微负坐标被映射到模拟盒中。

返回: 一个新的 Image 对象 (的列表)。每个对象包含超晶胞的一些信息。

例子:

```
from pwdata import Config, make_supercell

data_file = "./atom.config"
config = Config('pwmat/config', data_file)
supercell_matrix = [[2, 0, 0], [0, 2, 0], [0, 0, 2]]
supercell = make_supercell(config, supercell_matrix, pbc=[1, 1, 1])
supercell.to(data_path = "./", data_name = "atom_2x2x2.config",
↪   format = "pwmat/config", sort = True)
```

7.3.4 pwdata pertub.perturbation 微扰

Perturb the structure. 微扰结构

```
pertub.perturbation.perturb_structure (image_data, pert_num:int,
 ↪  cell_pert_fraction:float, atom_pert_distance:float)
```

参数:

- **image_data**: **必选**. 需要被微扰的 Image 对象，包含原始结构的一些信息。
- **pert_num**: 整数, **必选**. 要生成的微扰结构的数量。
- **cell_pert_fraction**: 浮点数, **必选**. 决定晶胞变形的程度。例如，`0.03` 表示晶胞变形的程度是相对原始晶胞的 3%。
- **atom_pert_distance**: 浮点数, **必选**. 原子微扰的距离，决定原子相对原始位置的移动距离。微扰是以埃为单位的距离。例如，`0.01` 表示原子的移动距离是 0.01 埃。

返回: 一个新的 Image 对象的列表。每个 Image 对象包含一个微扰结构的一些信息。

例子:

```
from pwdata import Config, perturb_structure

data_file = "./atom.config"
config = Config('pwmat/config', data_file)
pert_num = 50
cell_pert_fraction = 0.03
atom_pert_distance = 0.01
save_format = "pwmat/config"
perturbed_structs = perturb_structure(config, pert_num,
 ↪  cell_pert_fraction, atom_pert_distance)
perturbed_structs.to(data_path = "~/pwdata/test/pertubed/",
        data_name = "pertubed",
        format = save_format,
        direct = True,
        sort = True)
```

7.3.5 pwdata pertub.scale 缩放

```
pertub.scale.scale_cell (image_data, scale_factor:float)
```

参数:

- **image_data**: **必选**. 需要被缩放的 Image 对象，包含原始结构的一些信息。
- **scale_factor**: 浮点数, **必选**. 晶胞的缩放因子。

返回: 一个新的 Image 对象 (的列表)。每个 Image 对象包含一个缩放后的结构的一些信息。

例子:

```
from pwdata import Config, scale_cell

data_file = "./atom.config"
config = Config('pwmat/config', data_file)
scale_factor = 0.95
scaled_structs = scale_cell(config, scale_factor)
scaled_structs.to(data_path = "~/test/scaled/",
        data_name = "scaled",
        format = "pwmat/config",
        direct = True,
        sort = True)
```

7.3.6 pwdata 接口代码案例-把 MPtraj 文件转换为 lmdb 格式

MPtraj 开源数据集为 json 格式，转换为 lmdb 格式后，配合 cvt_configs 命令 -t 和 -q 选项 可以快速查找到指定结构，转换为 extxyz 或者 pwmlff/npy 格式做训练。

```
import json
from pwdata.fairchem.datasets.ase_datasets import LMDBDatabase
from ase import Atoms
from ase.db.row import AtomsRow
from pwdata.utils.constant import get_atomic_number_from_name
from tqdm import tqdm
import numpy as np

def MPjson2lmdb():
    mp_file =
 ↪  "/data/home/wuxingxing/codespace/pwdata/examples/mp_data/mptest.json"
    save_file =
 ↪  "/data/home/wuxingxing/codespace/pwdata/examples/mp_data/sub.aselmdb"
    Mpjson = json.load(open(mp_file))
    db = LMDBDatabase(filename=save_file, readonly=False)
    for key_1, val_1 in tqdm(Mpjson.items(),
     ↪  total=len(Mpjson.keys())):
        for key_2, val_2 in val_1.items():
            _atomrow, data = cvt_dict_2_atomrow(val_2)
            db._write(_atomrow, key_value_pairs={}, data=data)
    db.close()

def cvt_dict_2_atomrow(config:dict):
    cell = read_from_dict('matrix', config['structure']['lattice'],
 ↪  require=True)
    atom_type_list = get_atomic_number_from_name([_['label'] for _ in
 ↪  config['structure']['sites']])
```

```
    position = [_['xyz'] for _ in config['structure']['sites']]
    magmom = read_from_dict('magmom', config, require=True)
    atom = Atoms(positions=position,
                 numbers=atom_type_list,
                 magmoms=magmom,
                 cell=cell)

    atom_rows = AtomsRow(atom)
    atom_rows.pbc = np.ones(3, bool)
    # read stress -> xx, yy, zz, yz, xz, xy
    stress = read_from_dict('stress', config, require=True)
    atom_rows.stress =
↪   [stress[0][0],stress[1][1],stress[2][2],stress[1][2],stress[0][2],stress[0][[
    force = read_from_dict('force', config, require=True)
    energy = read_from_dict('corrected_total_energy', config,
↪   require=True)
    atom_rows.__setattr__('force',  force)
    atom_rows.__setattr__('energy', energy)
    data = {}
    data['uncorrected_total_energy'] =
↪   read_from_dict('uncorrected_total_energy', config, default=None)
    data['corrected_total_energy'] =
↪   read_from_dict('uncorrected_total_energy', config, default=None)
    data['energy_per_atom'] = read_from_dict('energy_per_atom',
↪   config, default=None)
    data['ef_per_atom'] = read_from_dict('ef_per_atom', config,
↪   default=None)
    data['e_per_atom_relaxed'] = read_from_dict('e_per_atom_relaxed',
↪   config, default=None)
    data['ef_per_atom_relaxed'] =
↪   read_from_dict('ef_per_atom_relaxed', config, default=None)
    data['bandgap'] = read_from_dict('bandgap', config, default=None)
    data['mp_id'] = read_from_dict('mp_id', config, default=None)
    return atom_rows, data

def read_from_dict(key:str, config:dict, default=None,
↪   require=False):
    if key in config:
        return config[key]
    else:
        if require:
            raise ValueError("key {} not found in config".format(key))
        else:
```

```
            return default
if __name__=="__main__":
    MPjson2lmdb()
```

8 MatPL 文献案例

在本章节，我们整理了使用 MatPL 做的一些测试工作，以及使用 MatPL 的已发表论文，为用户提供参考。

8.1 部分文献汇总

这里对案例中的文献做了整理总结，详情请参考相应文献对应的章节。

一、Comparison of Features

本例中比较了 MatPL 中的特征类型 在描述物理系统能力方面的差异。这些特征类型包括余弦特征、高斯特征、矩张量势 （MTP） 特征、光谱邻域分析势特征、具有切比雪夫多项式特征和高斯多项式特征的简化光滑深势以及原子簇展开特征。文章使用线性回归模型 在无定形硫体系和碳体系下，评估了针对密度泛函理论结果的原子群能量、总能量和力的训练均方根误差 (RMSE)。特征的详细介绍请参考Feature Wiki。

更多的测试细节也可以参考 龙讯公众号文章 以及 [文献 Accuracy evaluation of different machine learning force field features]

二、使用机器学习力场模拟液态硅到晶体硅的生长过程（案例）

[文献 Liquid to crystal Si growth simulation using machine learning force field]

本案例 使用 PMLFF 模拟了远离平衡态的硅熔体生长过程，发现基于第一性原理分解的原子能量 (PWmat 特性) 构建的 MLFF 可以准确再现第一性原理模拟的生长过程。提出了一种校正 ML-FF 训练中存在的系统偏差的方法，这对于准确模拟相变温度等关键结果非常重要。结果表明，ML-FF 可以准确模拟硅熔体生长过程，为使用 ML-FF 进行远离平衡态模拟提供了证据。

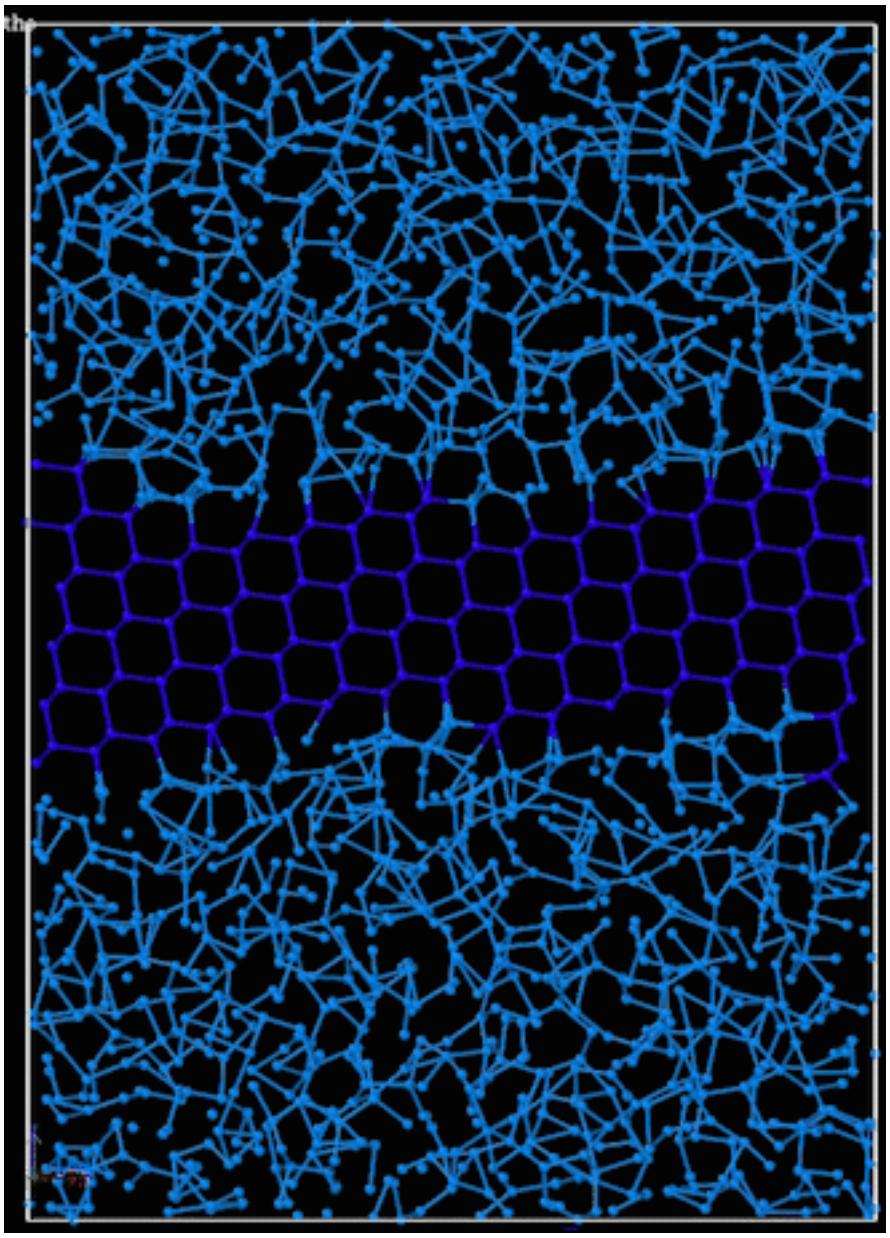


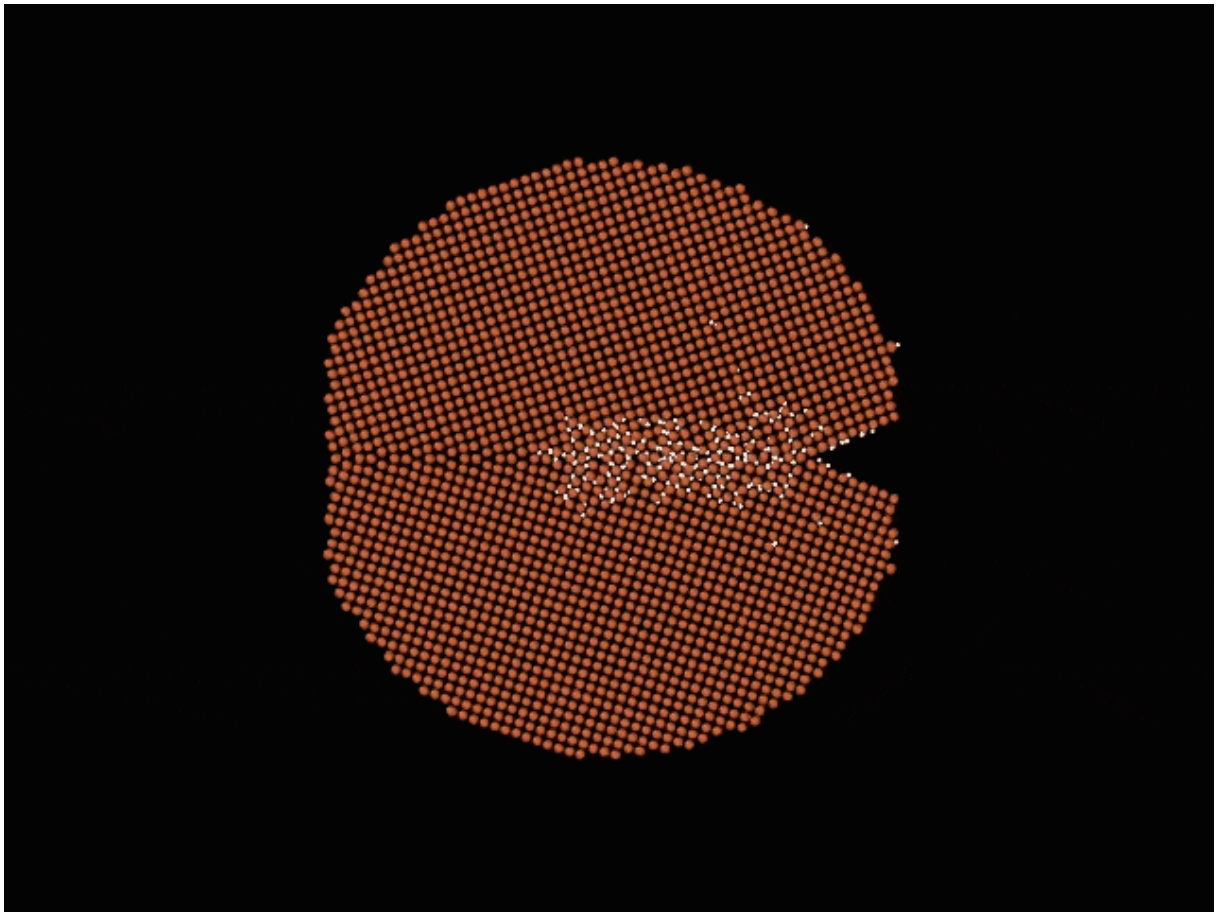
Figure 16: 硅生长过程

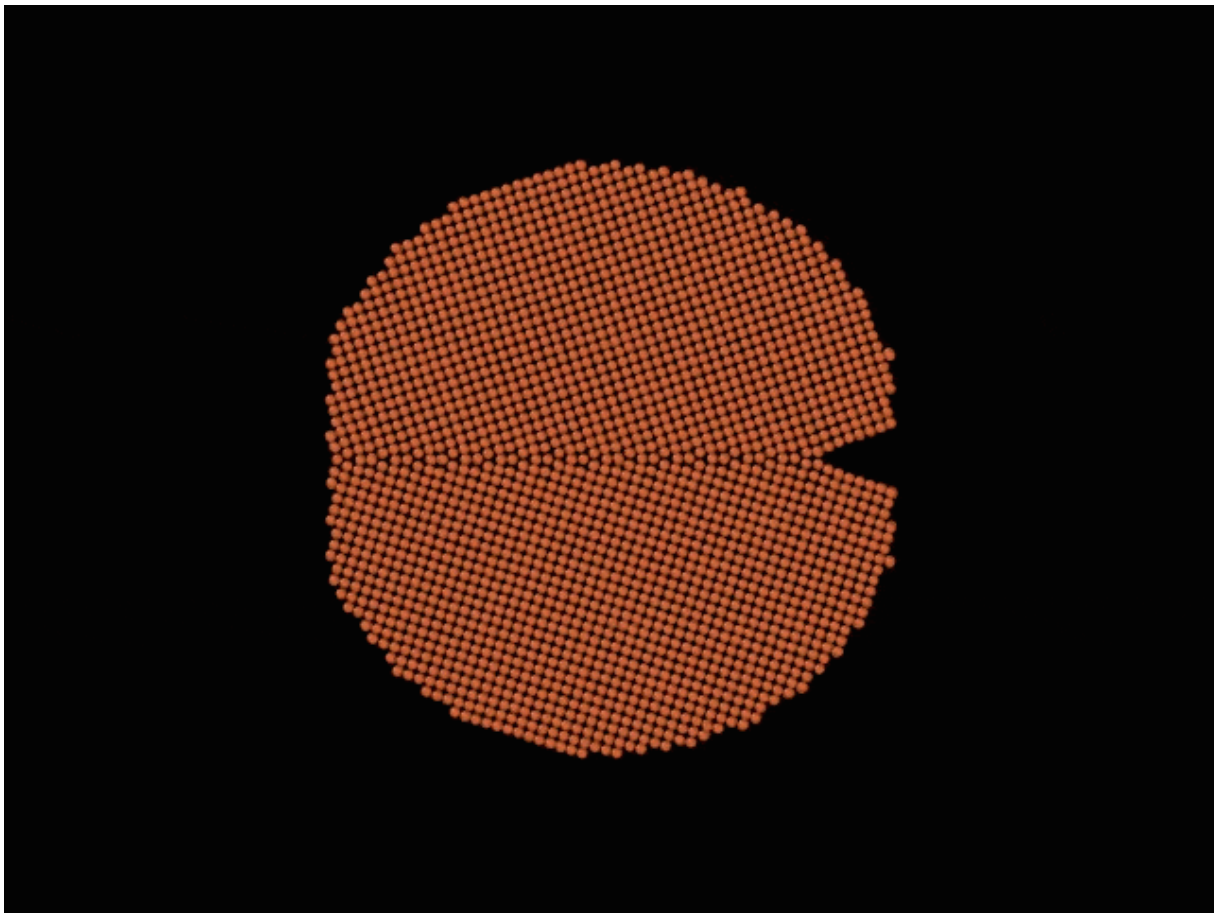
三、基于机器学习的铁-氢系统力场及其对 α-Fe 裂纹传播中氢的作用研究（案例）

[文献 Machine learning force field for Fe-H system and investigation on role of hydrogen on the crack propagation in α-Fe]

案例主要介绍了使用机器学习力场来研究氢对 α-铁裂纹传播影响的研究。具体内容包括：1. 基于密度泛函理论计算结果，构建了铁-氢体系的机器学习力场，该力场采用了神经网络模型，并训练了原子能量。该力场表现出了良好的统计和动力学性质。2. 通过分子动力学模拟，研究氢浓度对含裂纹的 α-铁模型试样裂纹传播的影响。研究发现氢浓度在裂纹尖端处越高，裂纹传播速度越快，这表明氢对裂纹的传播具有促进作用。3. 在含有晶界的试样中，观察到裂纹尖端处形成了微孔洞，这有助于释放裂纹尖端的拉伸应力，从而促进裂纹的传播。但微孔洞的形成似乎与氢的存在关系不大。4. 研究还发现，在 x 方向周期性较短的结构中，裂纹传播速度较快，这可能是由于 x 方向的协同效应导致的。5. 与嵌入原子力场的结果相比，机器学习力场显示出了

氢对裂纹传播的显著影响，这表明准确描述氢-金属相互作用的力场的重要性。6. 研究结果表明，氢在裂纹尖端聚集对氢脆性裂纹的传播起着关键作用，需要进一步深入研究不同条件下氢脆性的行为。





四、基于机器学习力场的分子动力学模拟揭示锂枝晶的形态演化（案例）

案例细节可以参考 [龙讯公众号文章] 以及 [文献 Revealing Morphology Evolution of Lithium Dendrites by Large-Scale Simulation Based on Machine Learning Force Field]

案例利用机器学习力场结合自洽连续溶剂模型，对锂枝晶在工作电解质环境中的形态演化进行了模拟。结果表明，枝晶形态演化可分为两个阶段。第一阶段由表面原子能量的降低驱动，导致原本单晶的枝晶发生局部取向重排，形成多个晶域。第二阶段由内部原子能量的降低驱动，促使晶域沿晶界滑移，并降低晶界能量。此外，文章还讨论了不同暴露表面取向对枝晶形态变化的影响。总的来说，降低表面能和晶界能驱动了枝晶形态的演化。

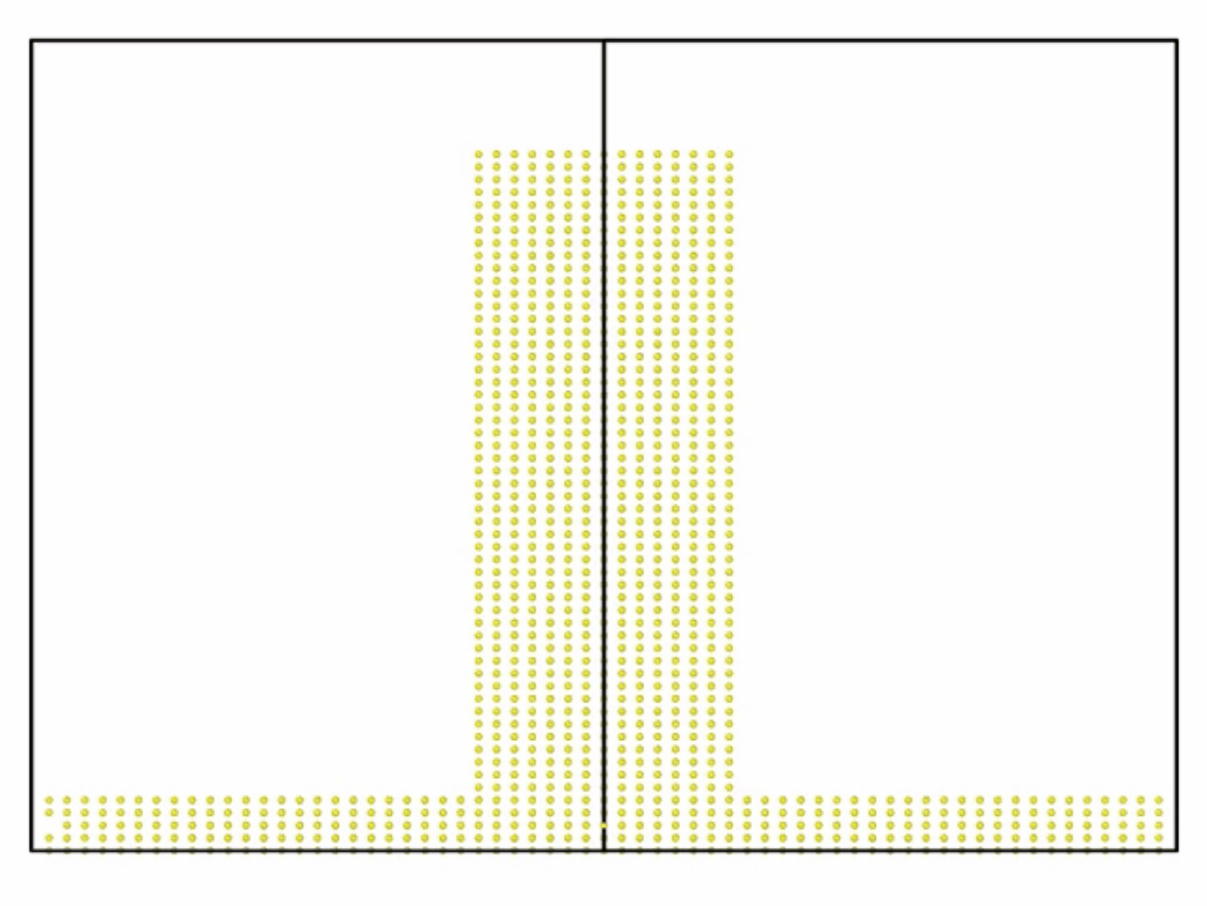


Figure 17: 锂枝晶在工作电解质环境中的形态演化

五、镁-铜合金力场（案例）

六、非晶硅机器学习力场计算非晶硅热导率（案例）

8.2 NN 特征值对比

[文章 Accuracy evaluation of different machine learning force field features]

这项工作比较了 MatPL 中的特征类型 在描述物理系统能力方面的差异。这些特征类型包括余弦特征、高斯特征、矩张量势 （MTP） 特征、光谱邻域分析势特征、具有切比雪夫多项式特征和高斯多项式特征的简化光滑深势以及原子簇展开特征。特征的详细介绍请参考Feature Wiki。

对于硫系统，在 300K 和 1500K 下进行了 NVT AIMD 模拟，并在 300K 进行了 2 ps 的分子动力学模拟。通过这些模拟获得了 2000 个 300K 下的结构。对于 1500K，进行了 3 ps 的分子动力学模拟，然后进行了 2 ps 的 AIMD 模拟，获得了 1500K 的训练数据集。硫环在模拟过程中破裂，呈现出了破裂键的系统结构。

对于碳系统，选择了 4 种不同的碳结构相，进行了从 300K 到 3500K 的 NVT AIMD 模拟。为了覆盖更广泛的构型空间，还在训练数据集中添加了 3500K 的高温模拟结果。每种碳相进行了 1000 步的模拟，共获得了 4000 个结构作为训练数据集。

System	Description	Temperature (K)	Steps (fs)
Sulfur-300 K	α-S 128 atoms	300	2000
Sulfur-1500 K	128 atoms	1500	2000
Diamond	64 atoms	300–3500	1000
Graphene	64 atoms	300–3500	1000
Graphenylene	64 atoms	300–3500	1000
M-carbon	64 atoms	300–3500	1000

硫和碳系统的细节及其 AIMD 步骤

部分实验结果

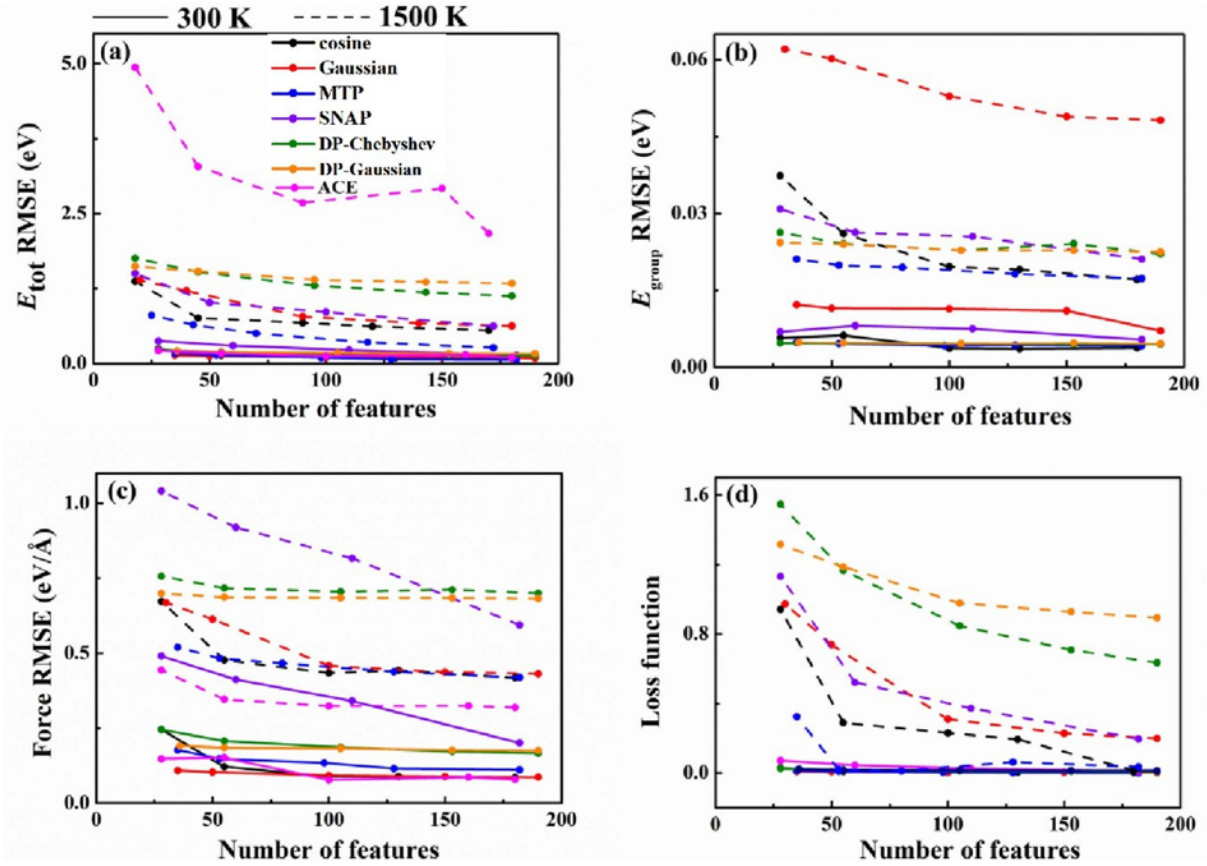


Figure 18: proportion_time

不同特征类型的硫-300 K 数据集 (实线) 和硫-1500 K 数据集 (虚线) 对 (a) 总能量、(b) 原子能量、(c) 力、(d) 损失函数的训练误差。

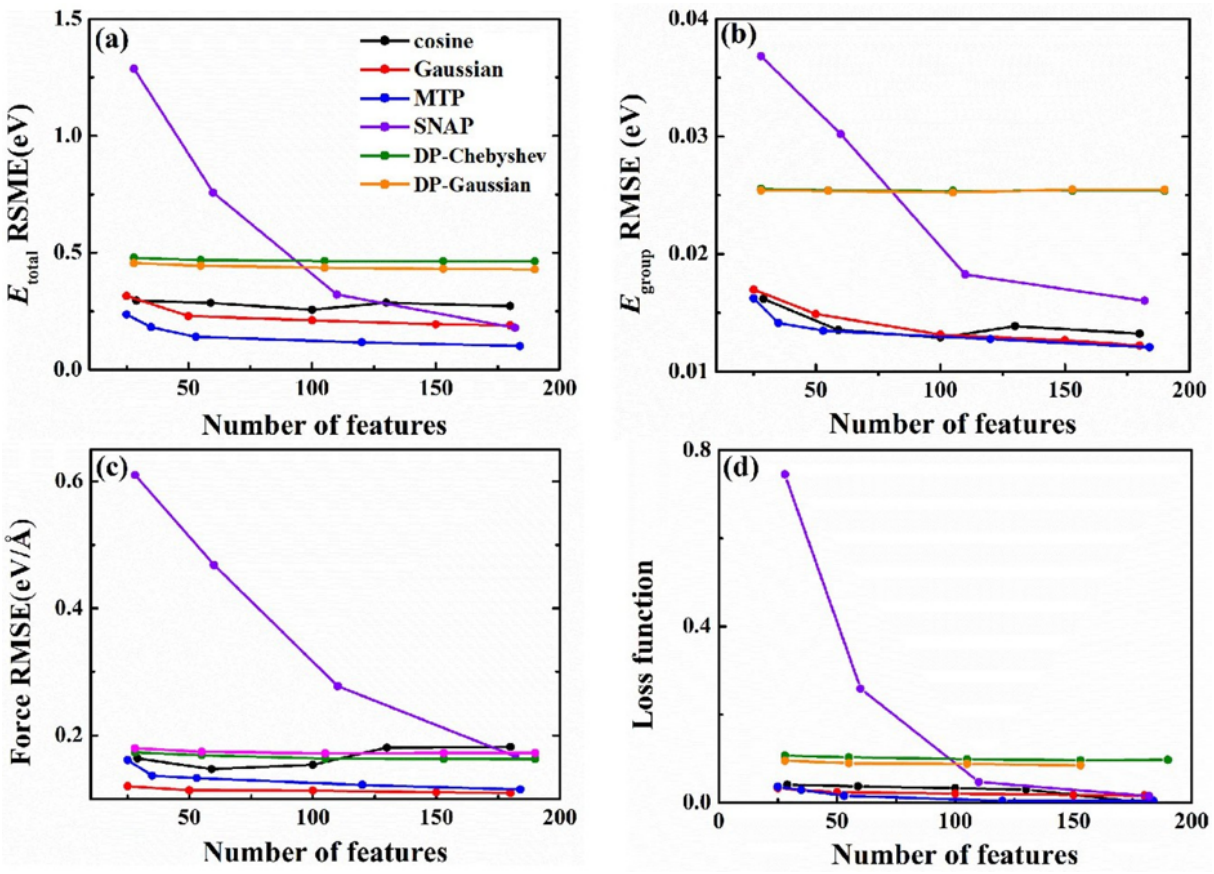


Figure 19: proportion_time

碳系统与不同特征类型组合数据集对 (a) 总能量、(b) 原子能量、(c) 力、(d) 损失函数的训练误差。

8.3 硅熔体生长过程

[文献 Si-growth]

案例 使用 PMLFF 模拟了远离平衡态的硅熔体生长过程，发现基于第一性原理分解的原子能量 (PWmat 特性) 构建的 MLFF 可以准确再现第一性原理模拟的生长过程。实验结果表明，MLFF 可以准确模拟硅熔体生长过程，为使用 MLFF 进行远离平衡态模拟提供了证据。

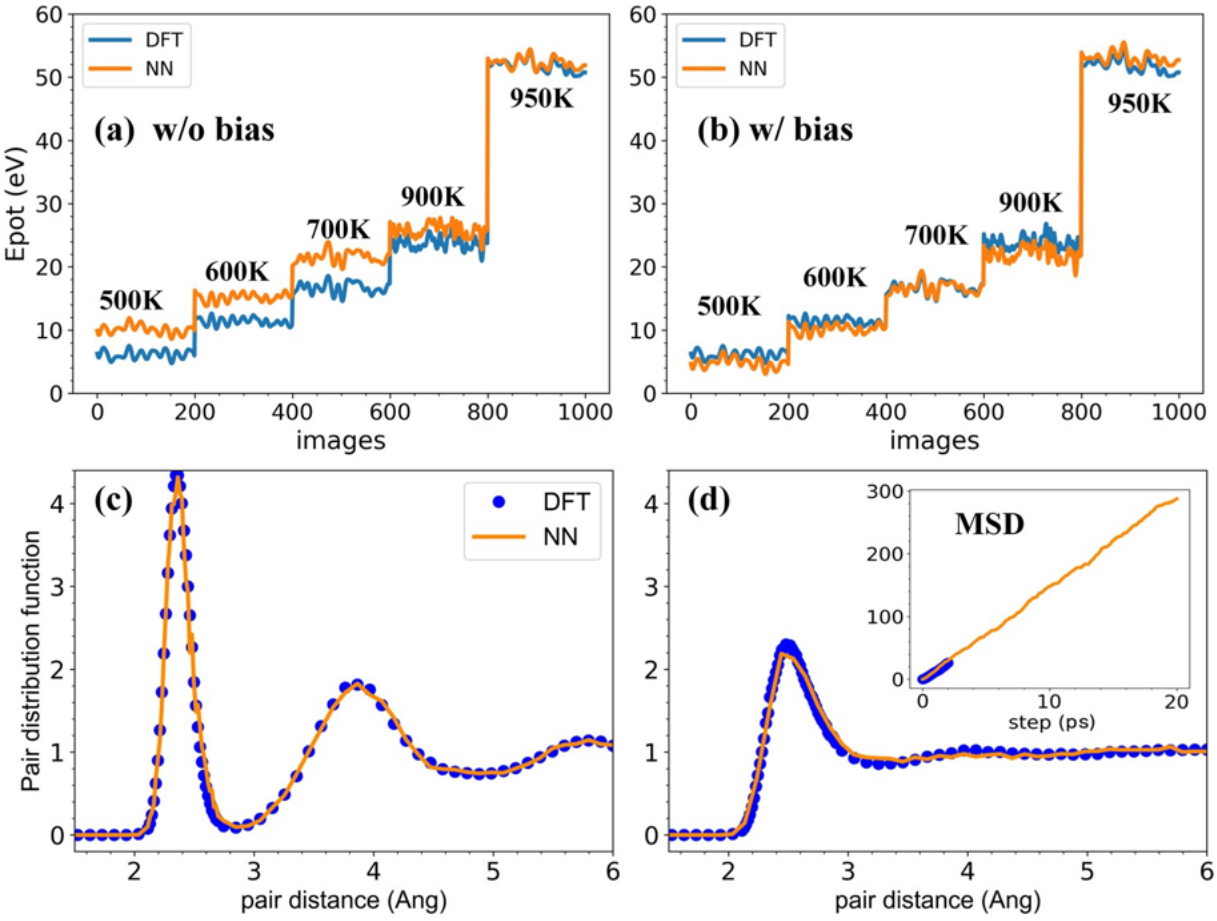


Figure 20: proportion_time

NN 和 DFT 的总能量对比，其中 (a) 没有偏差，(b) 有偏差。x 轴对应于不同温度下的晶体生长分子动力学（MD）模拟图像。(b) 的插图中是 1500 K 下液体相的均方位移；在 950 K 的晶体相（c）和 1500 K 的液体相（d）的配对分布函数。

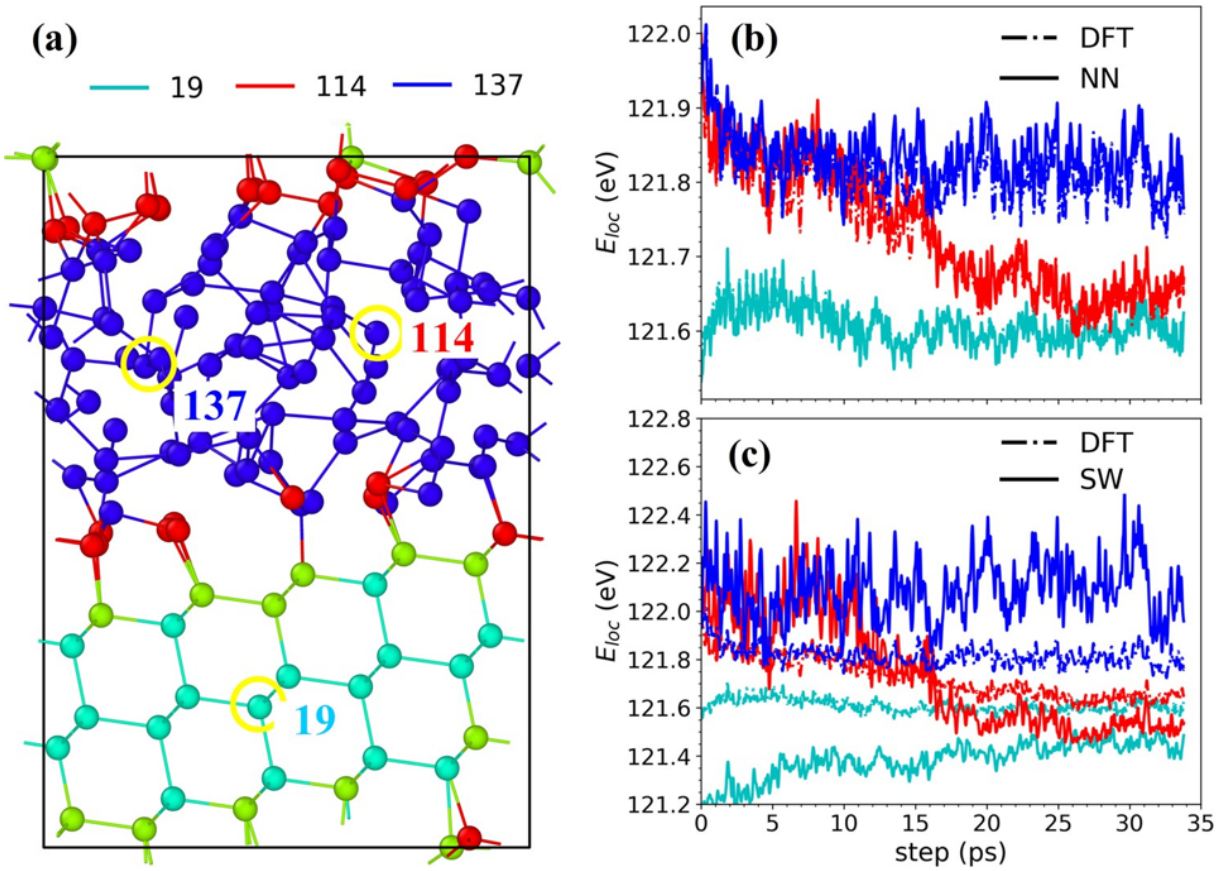


Figure 21: proportion_time

(a) 具有面心立方 (111) 表面的晶体硅薄片超胞结构，局部能量 Eloc(t) 的 DFT 与 (b) 神经网络和 SW 势能（Stillinger-Weber 经典力场）的比较。

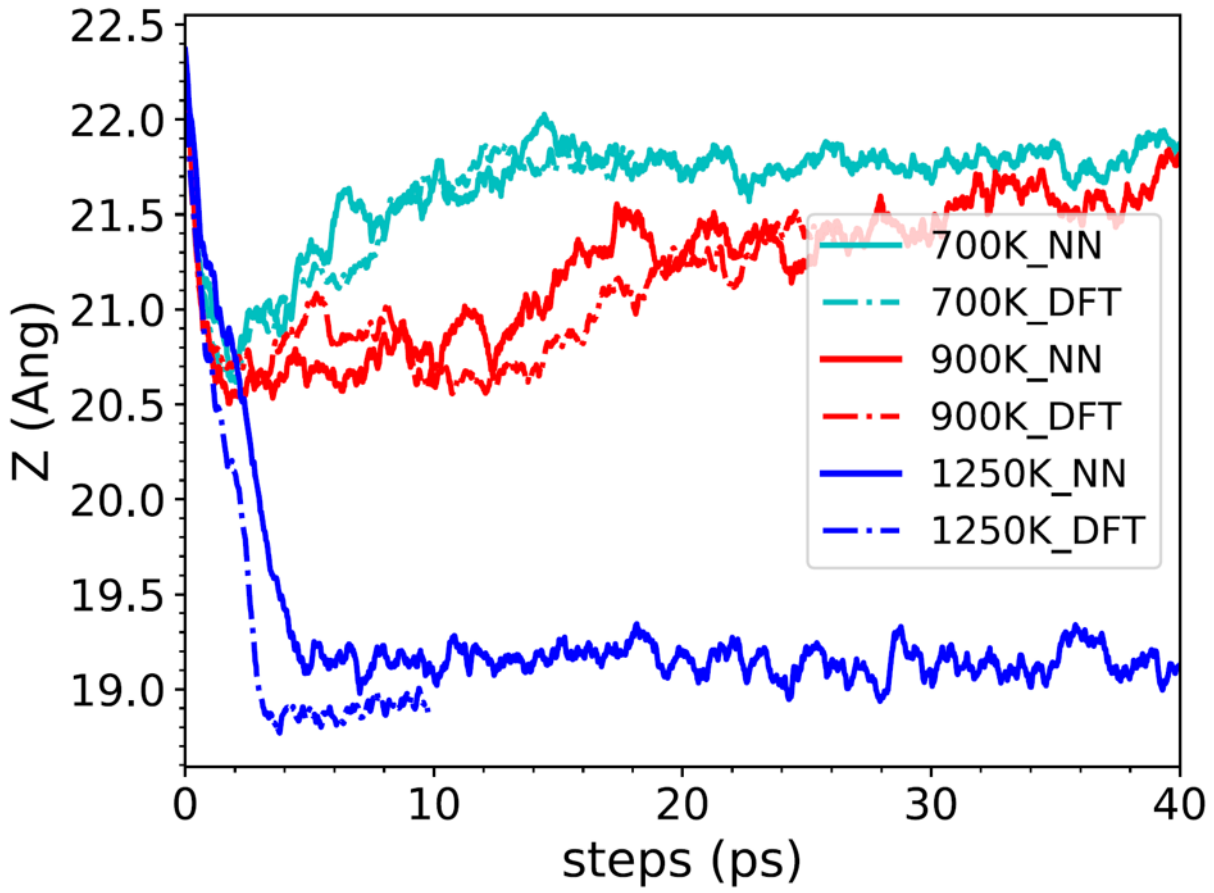


Figure 22: proportion_time

用 DFT 和 NN 模型（feature 1、2）计算 Z 长度随时间的增长。

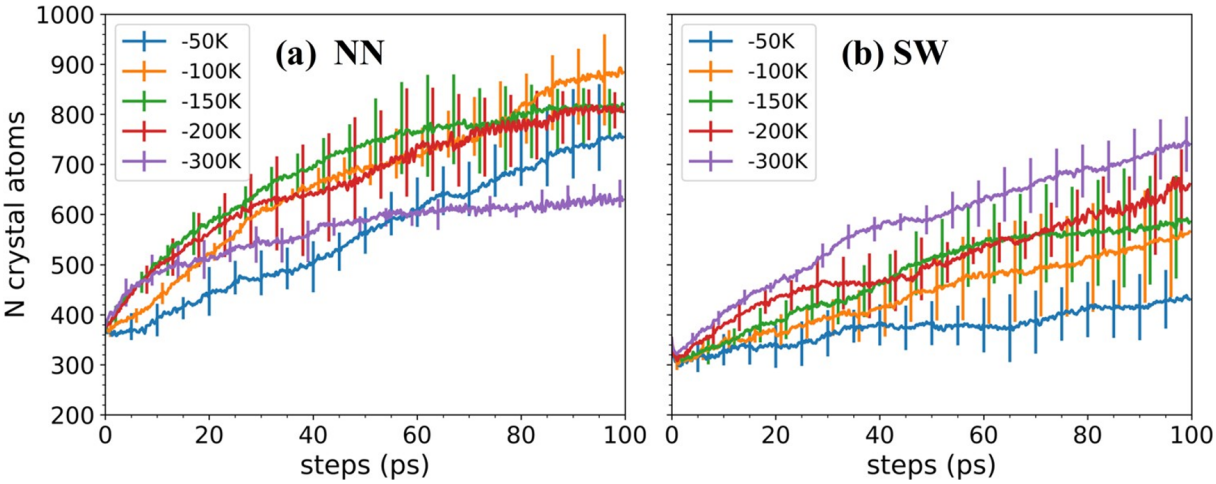


Figure 23: proportion_time

生长曲线（带误差棒）显示不同 ΔT 下的晶体原子数量：(a) 神经网络，(b)SW 势能。

8.4 铁-氢系统力场

[文献 Machine learning force field for Fe-H system and investigation on role of hydrogen on the crack propagation in α-Fe]

案例主要介绍了使用机器学习力场来研究氢对 α-铁裂纹传播影响的研究。具体内容包括：1. 基于密度泛函理论计算结果，构建了铁-氢体系的机器学习力场，该力场采用了神经网络模型，并训练了原子能量。该力场表现出了良好的统计和动力学性质。2. 通过分子动力学模拟，研究氢浓度对含裂纹的 α-铁模型试样裂纹传播的影响。研究发现氢浓度在裂纹尖端处越高，裂纹传播速度越快，这表明氢对裂纹的传播具有促进作用。3. 在含有晶界的试样中，观察到裂纹尖端处形成了微孔洞，这有助于释放裂纹尖端的拉伸应力，从而促进裂纹的传播。但微孔洞的形成似乎与氢的存在关系不大。4. 研究还发现，在 x 方向周期性较短的结构中，裂纹传播速度较快，这可能是由于 x 方向的协同效应导致的。5. 与嵌入原子力场的结果相比，机器学习力场显示出了氢对裂纹传播的显著影响，这表明准确描述氢-金属相互作用的力场的重要性。6. 研究结果表明，氢在裂纹尖端聚集对氢脆性裂纹的传播起着关键作用，需要进一步深入研究不同条件下氢脆性的行为。

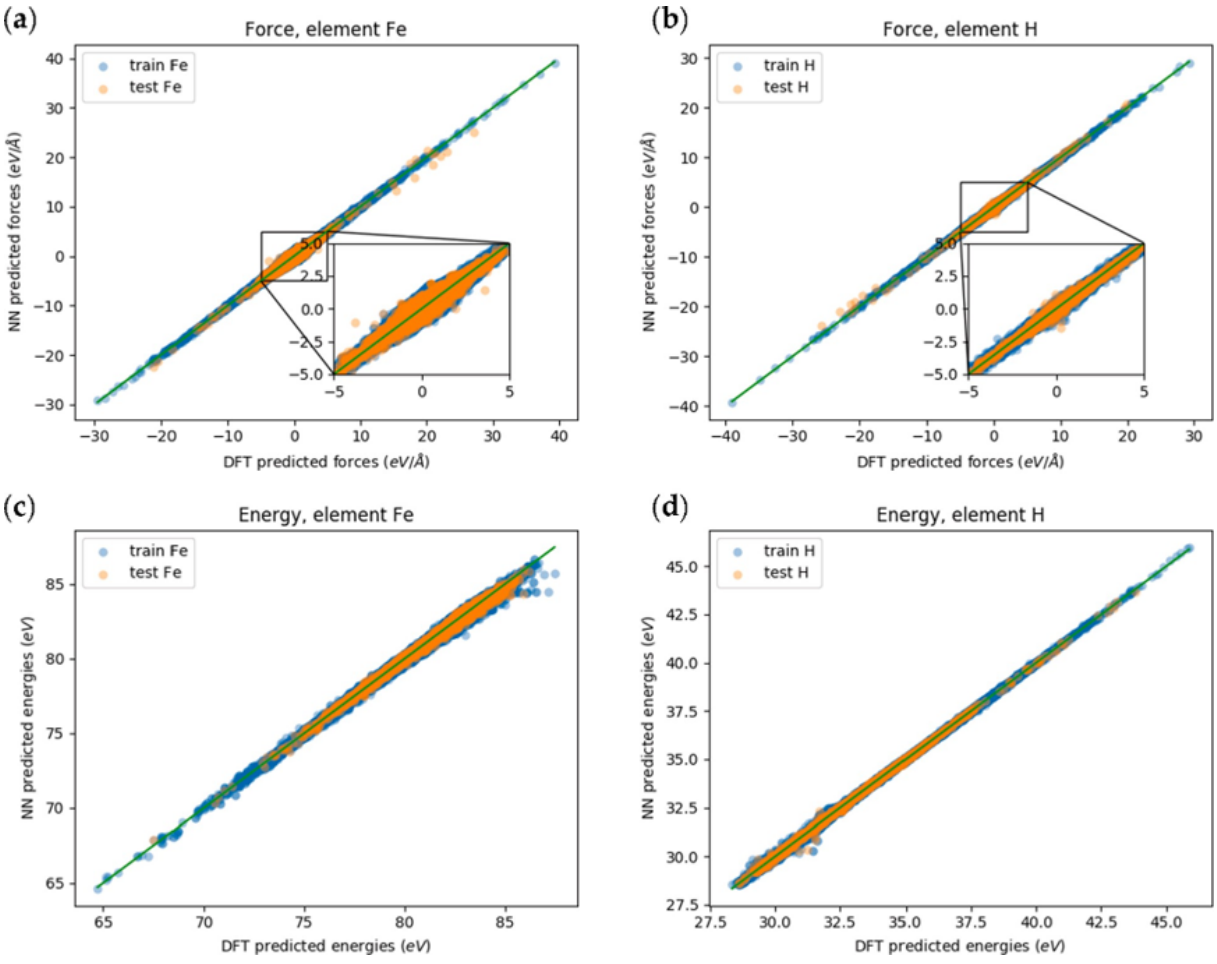


Figure 24: proportion_time

模型拟合精度

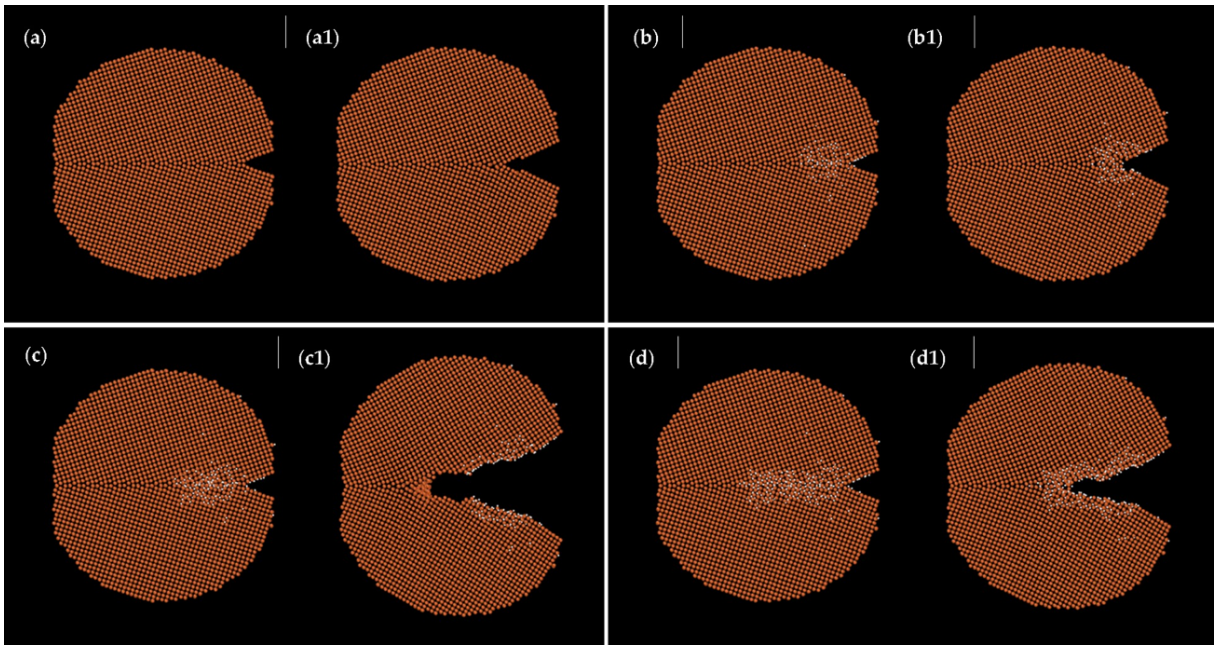


Figure 25: proportion_time

具有倾斜晶界的系统的分子动力学模拟。(a, a1) 无氢系统的初始和最终框架；(b, b1) 含 0.709% 总氢浓度的系统的初始和最终框架；(c, c1) 含 1.097% 总氢浓度的系统的初始和最终框架；(d, d1) 含 1.856% 总氢浓度的系统的初始和最终框架。

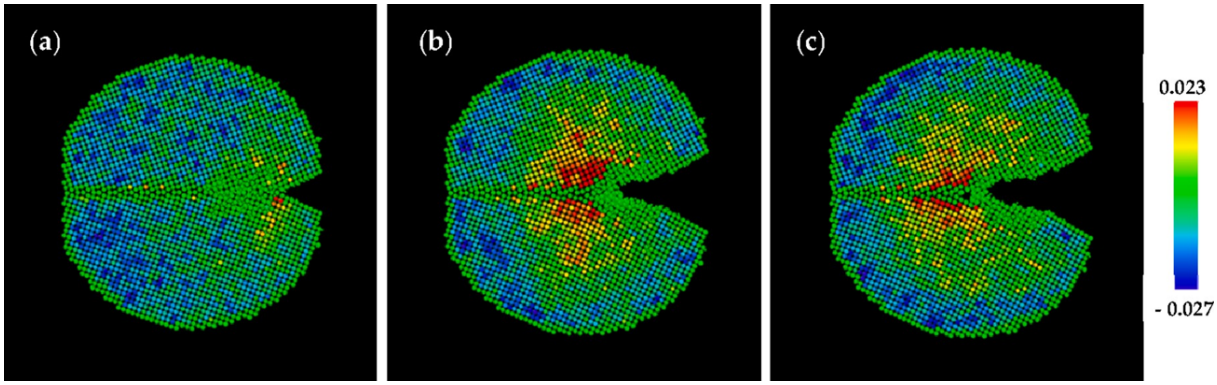


Figure 26: proportion_time

含有 1.097% 总氢浓度的系统的体积应变分布; (a) 初始框架; (b) 大约 6000 fs 时的框架。

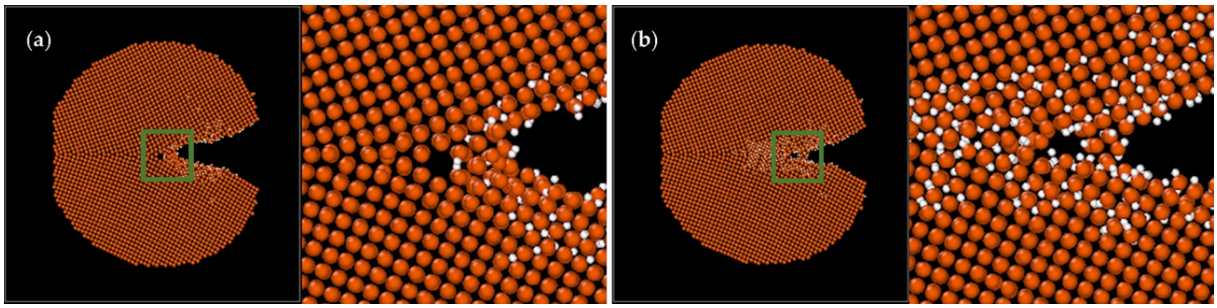


Figure 27: proportion_time

含有 1.097% 总氢浓度的系统的体积应变分布; (a) 初始框架; (b) 大约 6000 fs 的框架。

更多实验结果参考 文献 Machine learning force field for Fe-H system and investigation on role of hydrogen on the crack propagation in α-Fe

8.5 锂枝晶形态演化过程

案例细节可以参考 [龙讯公众号文章] 以及 [文献 Revealing Morphology Evolution of Lithium Dendrites by Large-Scale Simulation Based on Machine Learning Force Field]

锂离子电池具有高能量密度特性，在电动汽车、大规模储能系统等领域得到了广泛应用。锂金属负极由于同时具备最低化学势与高容量特性，有望成为下一代更高比能量密度电池中的重要组成部分。然而，在循环充放电中，锂金属负极易出现枝晶生长问题和体积剧烈变化问题，严重影响了锂金属负极的应用。在实际工作环境中，锂枝晶生长会大大降低电池的库仑效率、能量密度与稳定性。当枝晶生长到一定长度之后，还会刺穿隔膜，从而直接与正极接触，造成短路，可能导致火灾等严重事故。因此在锂金属负极材料的开发中，解决枝晶生长的问题至关重要。

随着实验技术的发展，锂枝晶的具体形态与相结构可由透射电子显微镜 (TEM) 直接观测。然而，受现有观测方法的分辨率限制，枝晶形态演化的动力学过程目前仍不甚清晰。现有的模拟研究则通常采用相场与经验力场方法，这些方法精度稍差，难以预测枝晶的实际形态特征，因此非常有必要发展原子级别精度的方法来做大规模形态模拟。

基于量子化学计算的机器学习方法提供了一个兼顾计算精度与速度的方案，这种机器学习力场模型通过训练高精度的小系统数据集得到，在碱金属以及其他材料上有望模拟到介观甚至宏观的尺度。此外，训练机器学习力场的势能面数据可以来自多种密度泛函理论计算，如隐式电解质环境中的锂原子计算，这使得人们有望在真实环境中模拟锂枝晶的演化过程。

本案例 采用基于 机器学习力场（MatPL）的分子动力学方法，模拟了在电解质环境中锂枝晶的形态演化过程，揭示了锂枝晶生长机制，对于推动锂金属负极材料的应用具有重要意义。

结合 DFT 精度原子能标签，加速模型构建

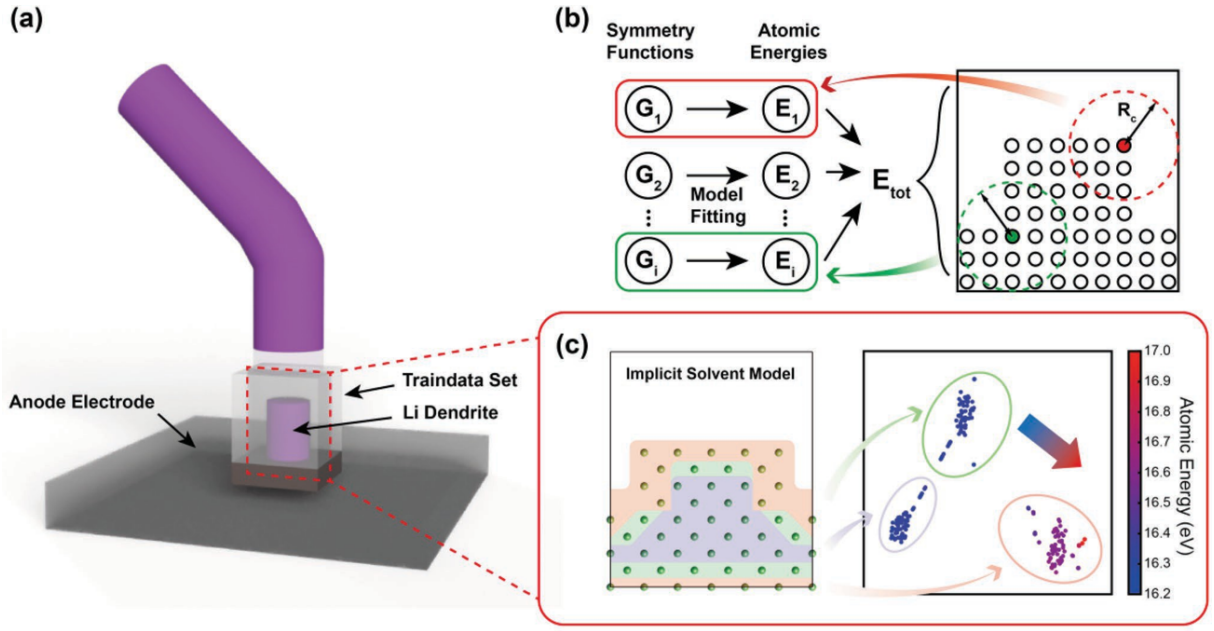


Figure 28: proportion_time

(a) 小尺度枝晶数据集构成（b）MLFF 模型结构（c）体心立方结构剖面图与特征-原子能关系可视化分析

针对跨尺度力场应用的主动学习与验证方案

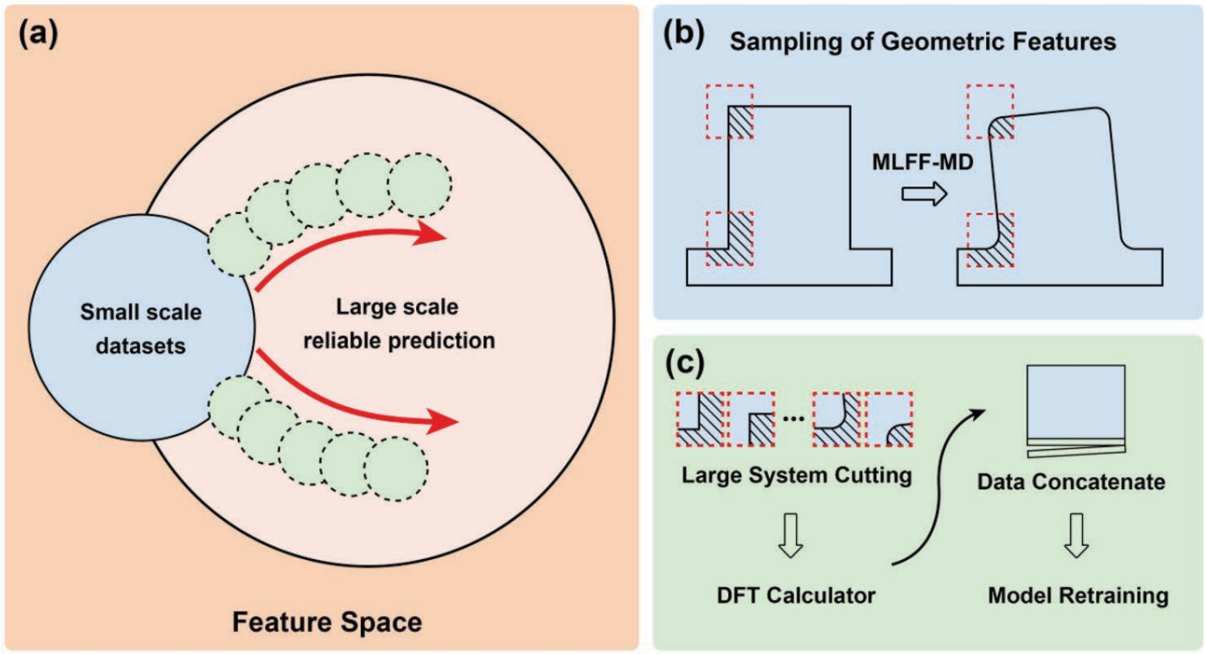


Figure 29: proportion_time

跨尺度模拟主动学习法示意图。（a）主动学习拓展模型中的数据采样示意图（b）机器学习力场-分子动力学中发生变化的关键部分截取采样（c）截取部分由 DFT 计算数据集重新训练模型

不同初始构型枝晶的形态演化过程与驱动力分析

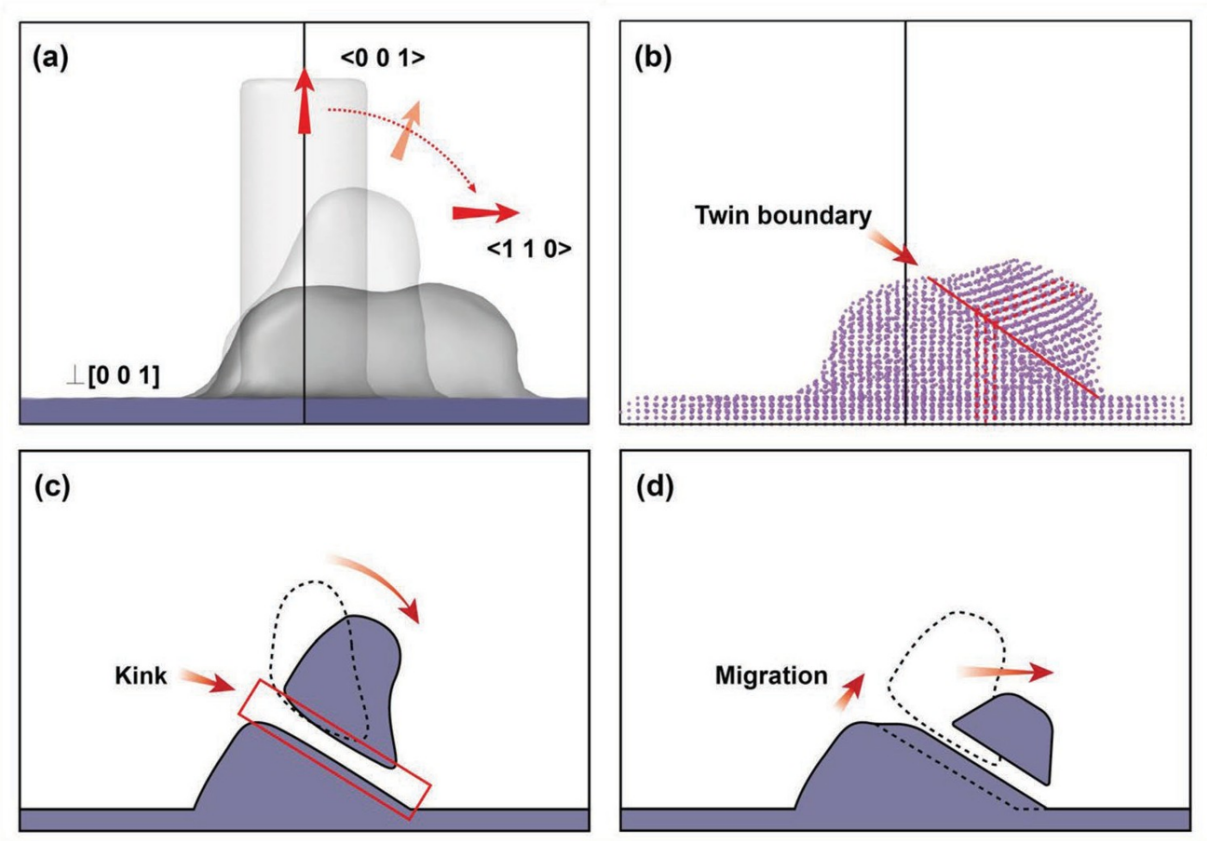


Figure 30: proportion_time

圆柱形结构的形态演化过程

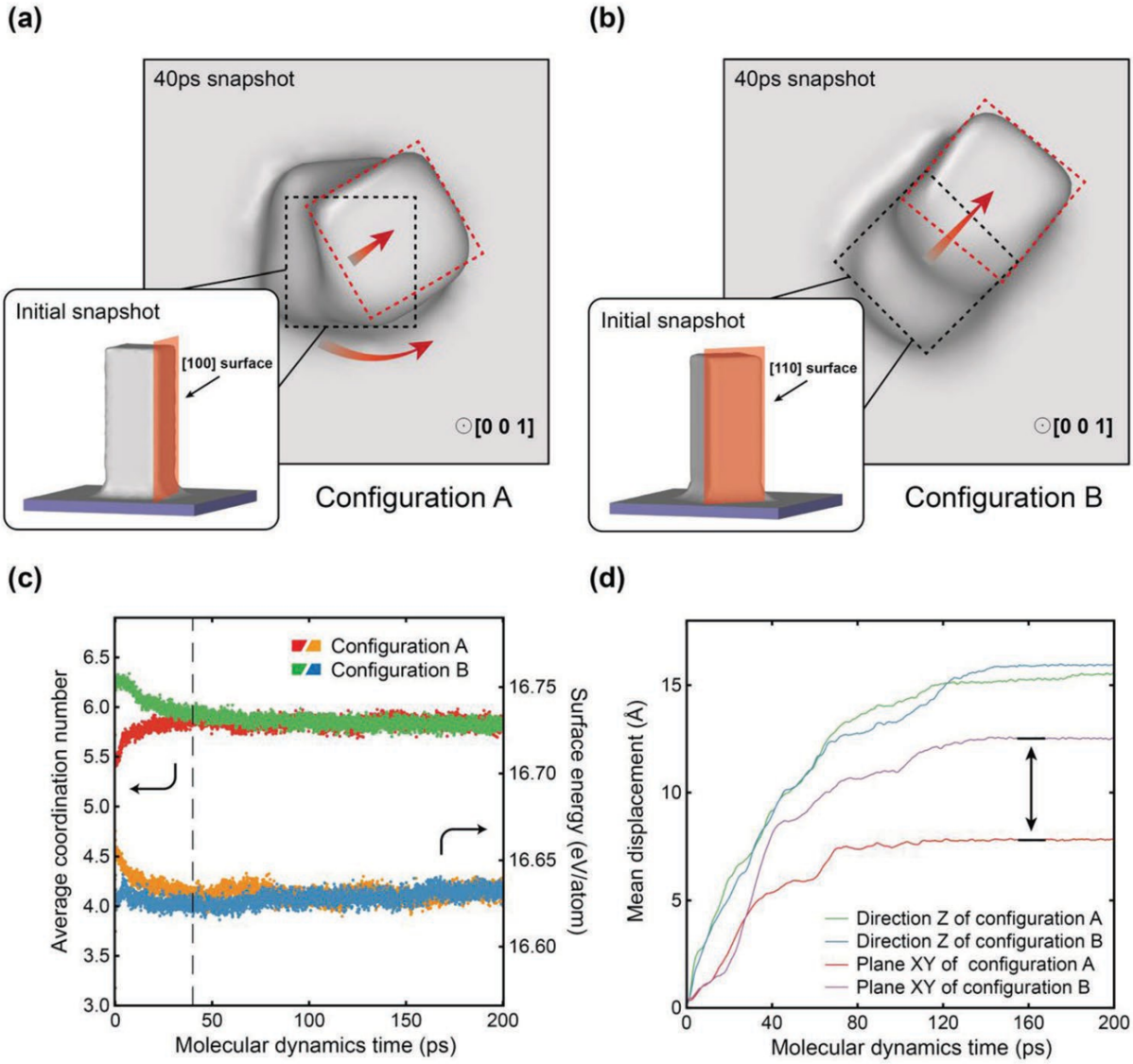


Figure 31: proportion_time

不同暴露面下矩形体结构的形态演化过程

8.6 镁-铜合金力场

Xingze Gen, Lin-Wang Wang, and Xiangying Meng. Comput. Mat. Sci. 246 (2025) 113486

拟合精度

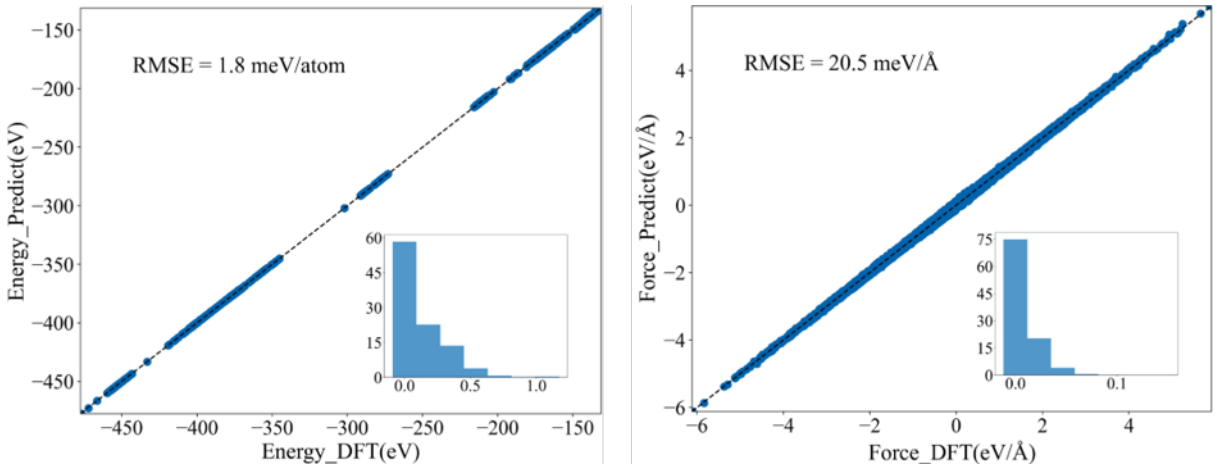


Figure 32: proportion_time

Mg、Cu 声子谱测试

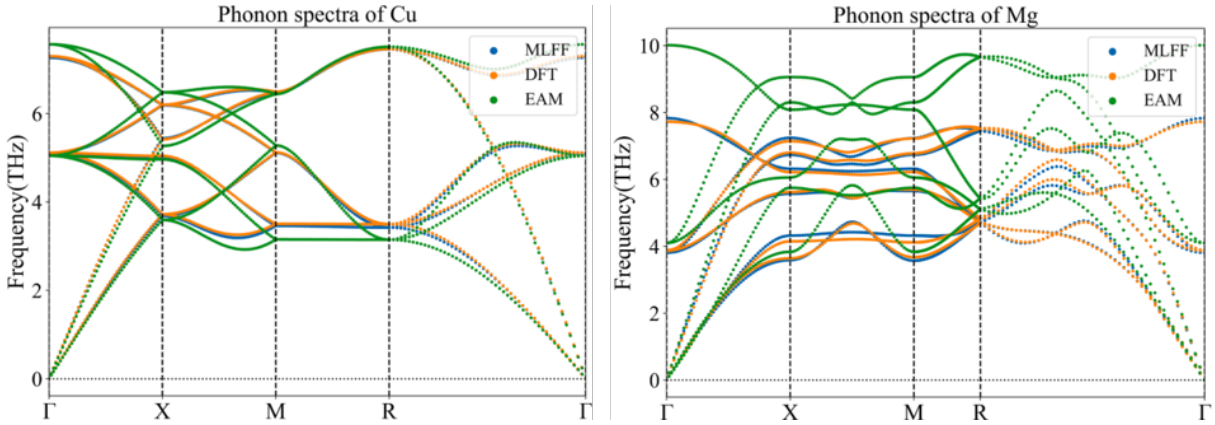


Figure 33: proportion_time

屈服强度

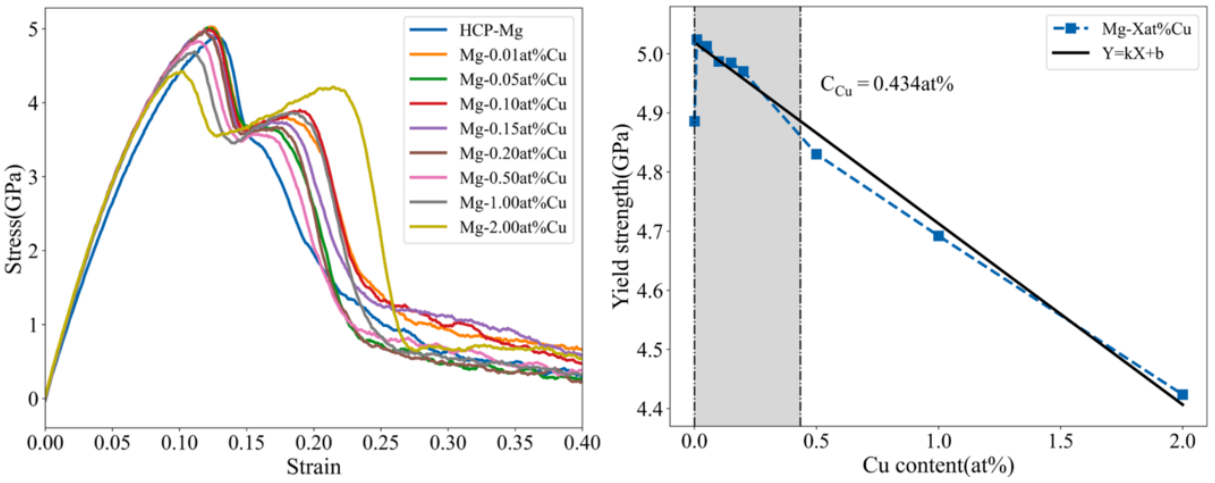


Figure 34: proportion_time

少量 Cu 含量可以提高合金的屈服强度，Cu 含量越高合金屈服强度越低

Mg-Cu 合金晶粒数对强度的影响

不同晶粒数结构建模

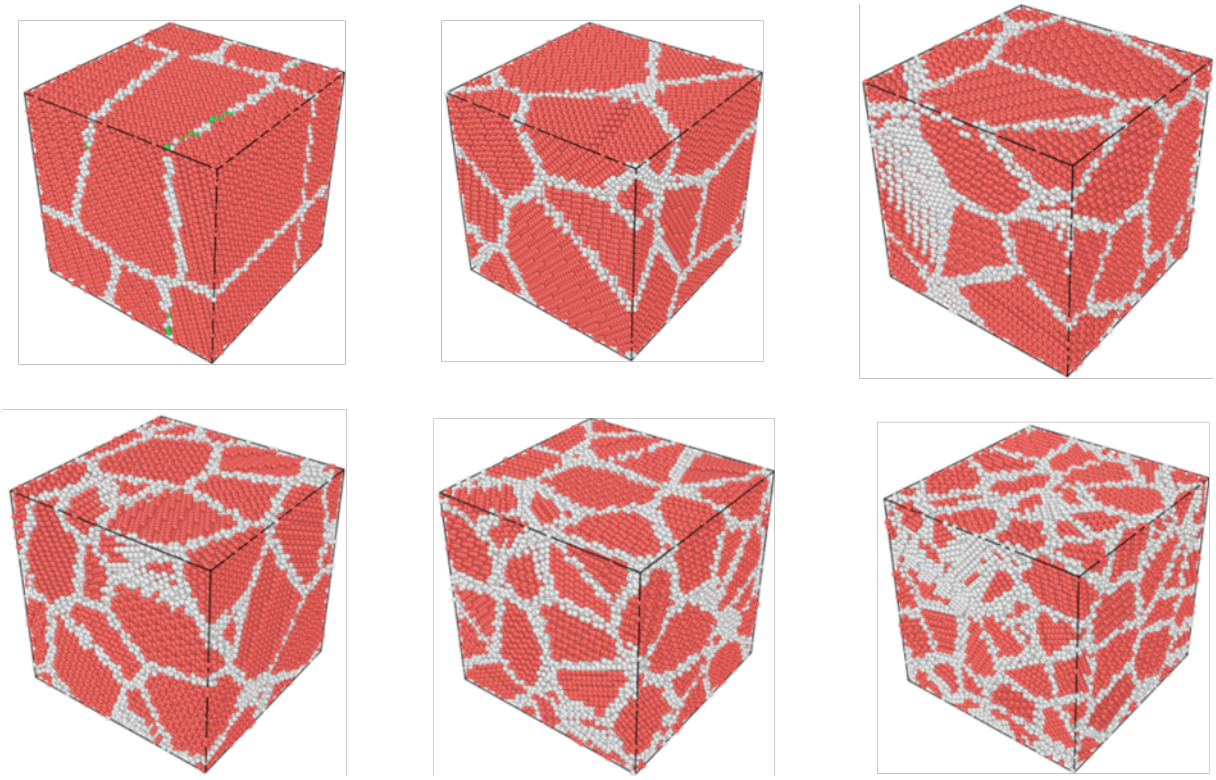


Figure 35: proportion_time

晶粒细化对合金强度影响较小

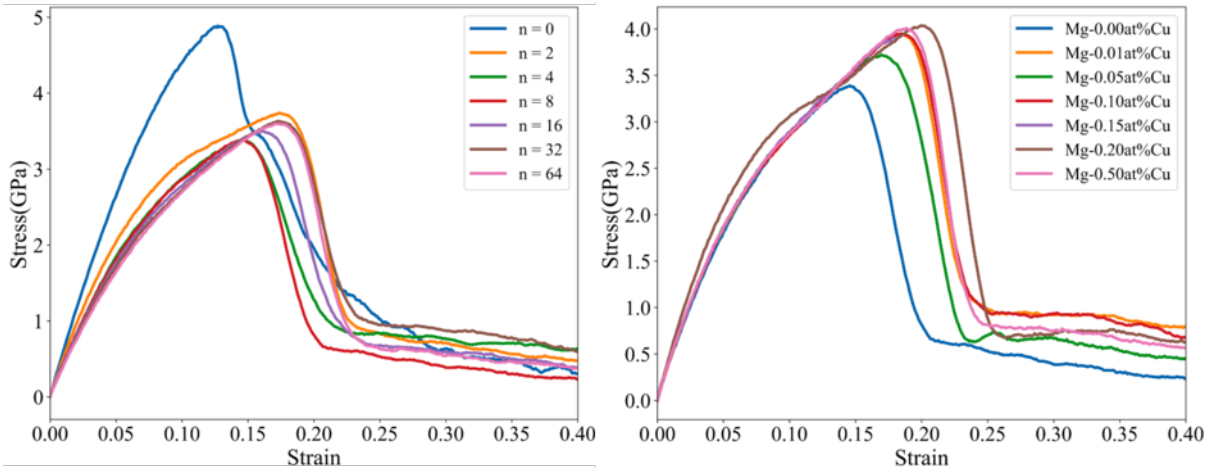


Figure 36: proportion_time

8.7 非晶硅机器学习力场计算非晶硅热导率

Yufeng Huang et al. Phys. Rev. B 99, 064103 (2019)

训练集精度

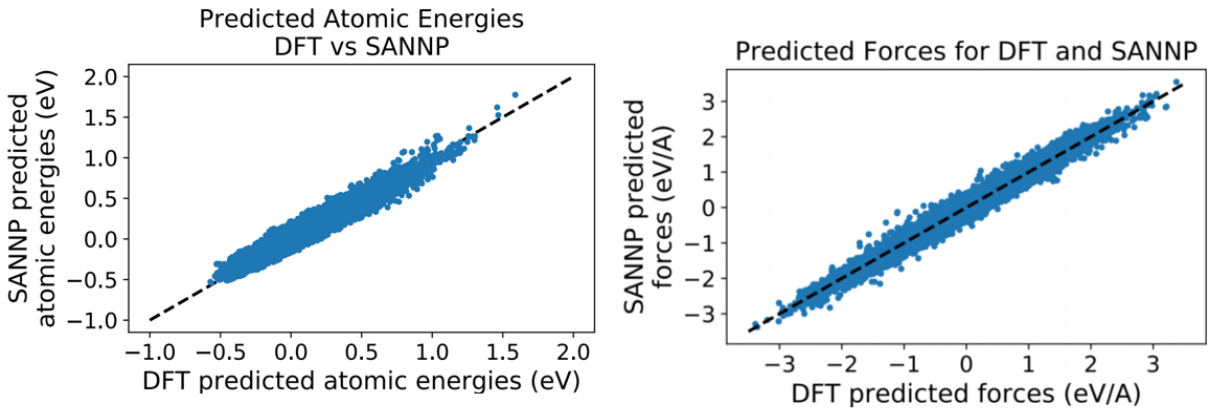


Figure 37: proportion_time

测试集精度

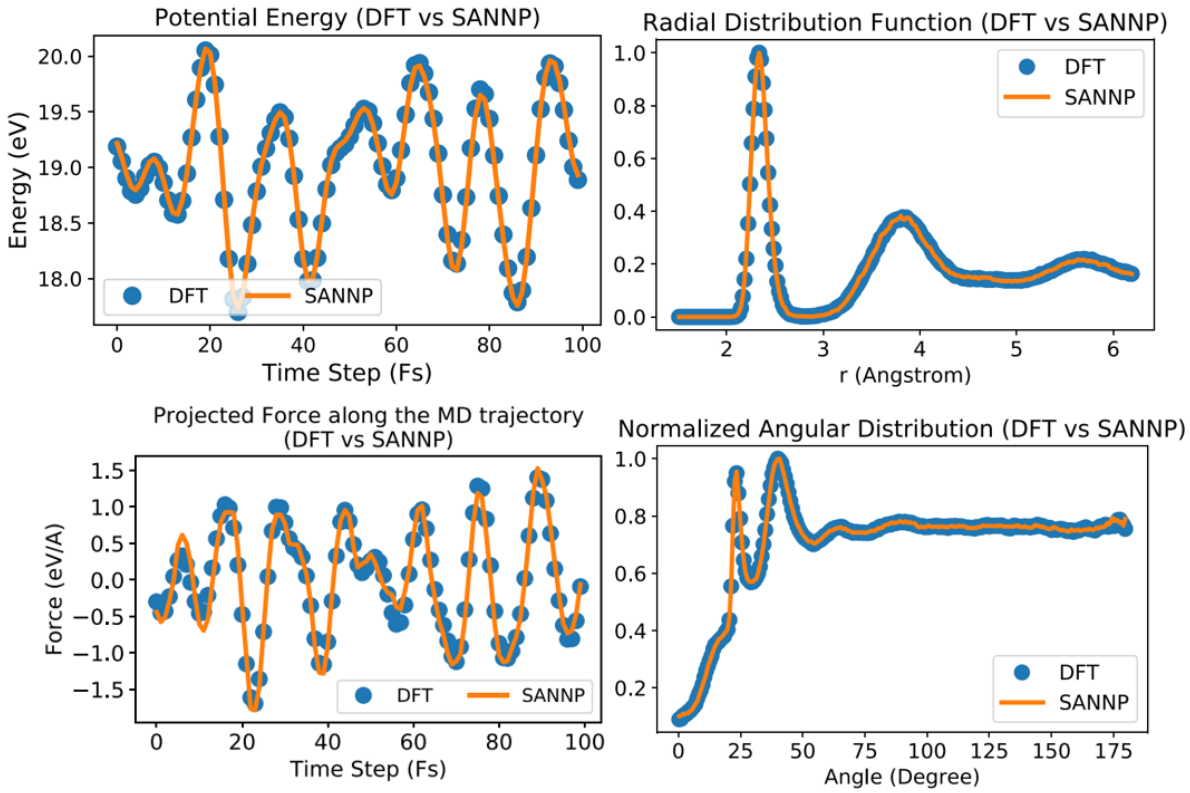


Figure 38: proportion_time

Green-Kubo 公式

$$\kappa_{\mu\nu}(t) = \frac{1}{k_B T^2 V} \int_0^t dt' \langle J_\mu(0) J_\nu(t') \rangle$$

$$\mathbf{J} = \sum_i \mathbf{v}_i E_i + \sum_i \mathbf{r}_i \frac{d}{dt} E_i$$

$$= \sum_i \mathbf{v}_i (K_i + U_i) + \sum_i \mathbf{W}_i \cdot \mathbf{v}_i$$

热导率

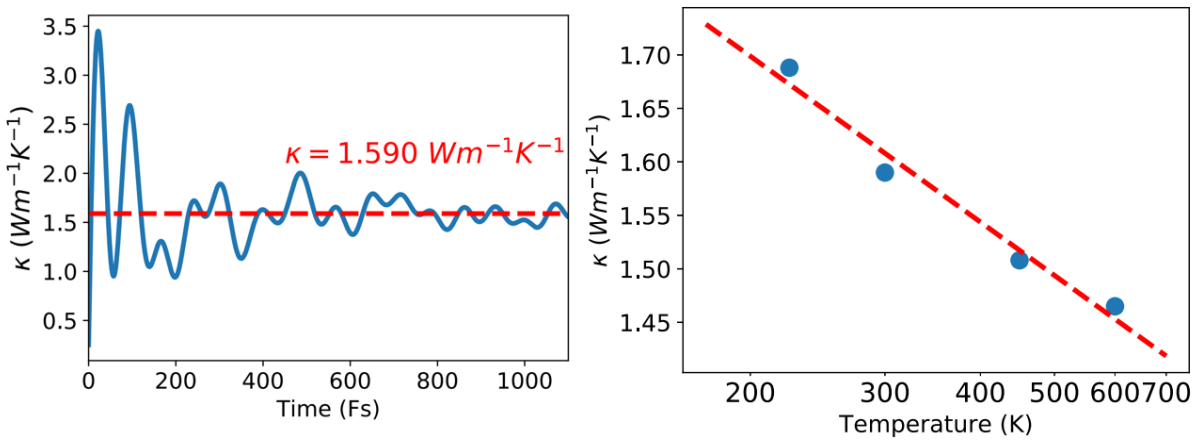


Figure 39: proportion_time