```
from __future__ import annotations

import torch
import sys
import os
import numpy as np
import pytorch_lightning as pl
from dgl.data.utils import split_dataset
from pytorch_lightning.loggers import CSVLogger

from matgl.ext.pymatgen import Structure2Graph, get_element_list
from matgl.graph.data import M3GNetDataset, MGLDataLoader,
collate_fn_efs
from matgl.models import M3GNet
from matgl.utils.training import PotentialLightningModule
from pymatgen.core import Structure
from get_db import Get_db

def set_seed(seed: int):
    torch.manual_seed(seed)
    np.random.seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed(seed)
        torch.cuda.manual_seed_all(seed)
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = False

if __name__ == '__main__':
    torch.set_default_device("cuda")
    # 加载已保存的图形数据
#    graphs, energies, forces = torch.load('saved_graphs.pt')

#    element_types = get_element_list(graphs)  # 使用图形数据获取元素
类型

    # 使用已加载的图形数据创建数据集

#    set_seed(543)
    dir_names = ['ran_data_vps_5A']
#    filename = 'ran_bcoh/351/vasprun.xml'
    strus = []
    energy = []
    forces = []
    stress = []
```

```
    for j in dir_names:
        for k in os.listdir(j):
            DB = Get_db()
            try:
                db                                                    =
DB.get_stru_energy_forces_stress(f"{j}/{k}/vasprun.xml")
                stru,e,f,stre = db
                k = int(len(stru)/50)
                strus += stru[::k]
                energy += e[::k]
                forces += f[::k]
                stress += stre
            except Exception as e:
                continue

    print(f"{len(strus)} found !!! .")

    element_types = ('H', 'B', 'C', 'O')
    converter = Structure2Graph(element_types=element_types, cutoff=5.0)
    dataset = M3GNetDataset(
        threebody_cutoff=4.0,
        structures=strus,
        converter=converter,
        energies=energy,
        forces=forces,
        stresses=None,
    )
    train_data, val_data, test_data = split_dataset(
        dataset,
        frac_list=[0.8, 0.2, 0.0],
        shuffle=True,
        random_state=345,
    )
    train_loader, val_loader, test_loader = MGLDataLoader(
        train_data=train_data,
        val_data=val_data,
        test_data=test_data,
        collate_fn=collate_fn_efs,
        batch_size=40,
        num_workers=0,
        generator=torch.Generator("cuda")
    )
    model = M3GNet(
        element_types=('H', 'B', 'C', 'O'),
```

```
        is_intensive=False,
        use_smooth=True
    )
    model = model.load('mgl.m3g')
    lit_model = PotentialLightningModule(model=model, lr=0.0001,
force_weight=10)

    trainer = pl.Trainer(max_epochs=500, accelerator='cuda', devices=1,
precision='32')
    trainer.fit(model=lit_model, train_dataloaders=train_loader,
val_dataloaders=val_loader)
    model.save(f'mgl.m3g_out')
```