In [ ]:
```
from __future__ import annotations

import os
import shutil
import warnings

import numpy as np
import pytorch_lightning as pl
from dgl.data.utils import split_dataset
from pymatgen.ext.matproj import MPRester
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
import torch

# To suppress warnings for clearer output
warnings.simplefilter("ignore")
```

In [ ]:
```
if torch.cuda.is_available():
  device = 'cuda'
else:
  device = 'cpu'

print(f'The available device is {device}')



#cuda = torch.device('cuda')
torch.set_default_device("cuda")
torch.backends.cuda.matmul.allow_tf32 = True
```

```
The available device is cuda
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('/home/charles/cp_mace/Chiku_data.csv', sep=';')
#df.head()
```

In [ ]:
```
df.info()
```

In [ ]:
```
import ase
from ase.io import read, write
from ase.build.tools import sort
from pymatgen.io.ase import AseAtomsAdaptor

structures = []
energies = []

vac1 = []
vac2 = []
for i in range(500):
    st = df.loc[:, 'POSCAR'][i]
```

```
    enegy = df.loc[:, 'total_energy'][i]
    ve1 = df.loc[:, 'vac_form_nrg_1'][i]
    ve2 = df.loc[:, 'vac_form_nrg_2'][i]
    file = open('POSCAR', 'w')
    file.write(st)
    file.close()
    st1 = read('POSCAR', format = 'vasp')
    #st1.info = {'energy': str(enegy)}
    #type(st1)
    sorted_st1 = sort(st1)
   # e=-100
    str = AseAtomsAdaptor.get_structure(sorted_st1)
    structures.append(str)

    #energies.append(ve1)
    energies.append(enegy)
    vac1.append(ve1)
    vac2.append(ve2)
    #write('t.xyz', st1, format = 'extxyz')
#ase.io.extxyz.write_extxyz('b.test.2.xyz', structures)



forces = [np.zeros((len(s), 3)).tolist() for s in structures]
stresses = [np.zeros((3, 3)).tolist() for s in structures]
```

In [ ]:
```
print(type(structures[0]))
print(len(structures))
print(len(forces))
print(len(energies))
print(len(stresses))
```

```
<class 'pymatgen.core.structure.Structure'>
500
500
500
500
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.rcParams.update({'figure.figsize':(7,5), 'figure.dpi':100})

# Plot Histogram on x
x = energies

plt.hist(x, bins=50)
plt.gca().set(title='Frequency Histogram', ylabel='Frequency');
```

In [ ]:
```
# Plot Histogram on x
y = vac1

plt.hist(y, bins=50)
plt.gca().set(title='Frequency Histogram', ylabel='Frequency');
```

In [ ]:
```
# Plot Histogram on x
z = vac2
```

```
plt.hist(z, bins=50)
plt.gca().set(title='Frequency Histogram', ylabel='Frequency');
```

In [ ]:

In [ ]:
```
'''
mpr = MPRester('ExgOF1NIKcWWxTrvUD487pmztYWZQg3m')

entries = mpr.get_entries_in_chemsys(["Si", "O"])
structures = [e.structure for e in entries]
energies = [e.energy for e in entries]
forces = [np.zeros((len(s), 3)).tolist() for s in structures]
stresses = [np.zeros((3, 3)).tolist() for s in structures]

print(f"{len(structures)} downloaded from MP.")

'''
```

In [ ]:
```
element_types = get_element_list(structures)
converter = Structure2Graph(element_types=element_types, cutoff=5.0)
dataset = M3GNetDataset(
    threebody_cutoff=4.0,
    structures=structures,
    converter=converter,
    energies=energies,
    forces=forces,
    stresses=stresses,
)
train_data, val_data, test_data = split_dataset(
    dataset,
    frac_list=[0.8, 0.1, 0.1],
    shuffle=True,
    random_state=42,
)
train_loader, val_loader, test_loader = MGLDataLoader(
    train_data=train_data,
    val_data=val_data,
    test_data=test_data,
    num_workers=0,
    collate_fn=collate_fn_efs,
    generator='cuda', #None,
    batch_size=32,
)
model = M3GNet(
    element_types=element_types,
    is_intensive=False,
)


lr = 1e-4
lit_module = PotentialLightningModule(model=model, lr=lr)# force_weight=0.00
```

```
100%|██████████| 500/500 [00:21<00:00, 23.08it/s]
```

In [ ]:
```
!nvidia-smi
```

```
Mon Aug  7 20:15:33 2023
+-----------------------------------------------------------------------------
--+
| NVIDIA-SMI 520.61.05    Driver Version: 520.61.05    CUDA Version: 11.8
|
|-------------------------------+----------------------+----------------------
--+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. EC
C |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute
M. |
|                               |                      |               MIG
M. |
|===============================+======================+======================
==|
|   0  NVIDIA RTX A500...  On   | 00000000:01:00.0  On |                  N/
A |
| N/A   57C    P3    24W /  N/A |    603MiB / 16384MiB |      5%      Defaul
t |
|                               |                      |                  N/
A |
+-------------------------------+----------------------+----------------------
--+

+-----------------------------------------------------------------------------
--+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                  GPU Memor
y |
|        ID   ID                                                   Usage
|
|=============================================================================
==|
|    0   N/A  N/A      2254      G   /usr/lib/xorg/Xorg                 95Mi
B |
|    0   N/A  N/A      5316      C   ...da3/envs/matgl/bin/python      504Mi
B |
+-----------------------------------------------------------------------------
--+
```

In [ ]:
```
# If you wish to disable GPU or MPS (M1 mac) training, use the accelerator="
logger = CSVLogger("logs", name="b.perfect_structure_test")
trainer = pl.Trainer(max_epochs=100, accelerator="cuda", logger=logger) #, a
trainer.fit(model=lit_module, train_dataloaders=train_loader, val_dataloader
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name  | Type              | Params
-------------------------------------------
0 | mae   | MeanAbsoluteError | 0
1 | rmse  | MeanSquaredError  | 0
2 | model | Potential         | 283 K
-------------------------------------------
283 K     Trainable params
0         Non-trainable params
283 K     Total params
1.133     Total estimated model params size (MB)
Sanity Checking: 0it [00:00, ?it/s]
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[8], line 4
      2 logger = CSVLogger("logs", name="b.perfect_structure_test")
      3 trainer = pl.Trainer(max_epochs=100, accelerator="cuda", logger=logg
er) #, accelerator="cpu")
----> 4 trainer.fit(model=lit_module, train_dataloaders=train_loader, val_da
taloaders=val_loader)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/t
rainer/trainer.py:529, in Trainer.fit(self, model, train_dataloaders, val_da
taloaders, datamodule, ckpt_path)
    527 model = _maybe_unwrap_optimized(model)
    528 self.strategy._lightning_module = model
--> 529 call._call_and_handle_interrupt(
    530     self, self._fit_impl, model, train_dataloaders, val_dataloaders,
datamodule, ckpt_path
    531 )

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/t
rainer/call.py:42, in _call_and_handle_interrupt(trainer, trainer_fn, *args,
**kwargs)
     40     if trainer.strategy.launcher is not None:
     41         return trainer.strategy.launcher.launch(trainer_fn, *args, t
rainer=trainer, **kwargs)
---> 42     return trainer_fn(*args, **kwargs)
     44 except _TunerExitException:
     45     _call_teardown_hook(trainer)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/t
rainer/trainer.py:568, in Trainer._fit_impl(self, model, train_dataloaders,
val_dataloaders, datamodule, ckpt_path)
    558 self._data_connector.attach_data(
    559     model, train_dataloaders=train_dataloaders, val_dataloaders=val_
dataloaders, datamodule=datamodule
    560 )
    562 ckpt_path = self._checkpoint_connector._select_ckpt_path(
    563     self.state.fn,
    564     ckpt_path,
    565     model_provided=True,
    566     model_connected=self.lightning_module is not None,
    567 )
--> 568 self._run(model, ckpt_path=ckpt_path)
    570 assert self.state.stopped
    571 self.training = False

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/t
rainer/trainer.py:973, in Trainer._run(self, model, ckpt_path)
    968 self._signal_connector.register_signal_handlers()
    970 # ----------------------------
    971 # RUN THE TRAINER
    972 # ----------------------------
--> 973 results = self._run_stage()
    975 # ----------------------------
    976 # POST-Training CLEAN UP
    977 # ----------------------------
```

```
    978 log.debug(f"{self.__class__.__name__}: trainer tearing down")

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/trainer/trainer.py:1014, in Trainer._run_stage(self)
   1012 if self.training:
   1013     with isolate_rng():
-> 1014         self._run_sanity_check()
   1015     with torch.autograd.set_detect_anomaly(self._detect_anomaly):
   1016         self.fit_loop.run()

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/trainer/trainer.py:1043, in Trainer._run_sanity_check(self)
   1040 call._call_callback_hooks(self, "on_sanity_check_start")
   1042 # run eval step
-> 1043 val_loop.run()
   1045 call._call_callback_hooks(self, "on_sanity_check_end")
   1047 # reset logger connector

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/loops/utilities.py:177, in _no_grad_context.<locals>._decorator(self, *args, **kwargs)
    175     context_manager = torch.no_grad
    176 with context_manager():
--> 177     return loop_run(self, *args, **kwargs)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/loops/evaluation_loop.py:115, in _EvaluationLoop.run(self)
    113     previous_dataloader_idx = dataloader_idx
    114     # run step hooks
--> 115     self._evaluation_step(batch, batch_idx, dataloader_idx)
    116 except StopIteration:
    117     # this needs to wrap the `*_step` call too (not just `next`) for `dataloader_iter` support
    118     break

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/loops/evaluation_loop.py:375, in _EvaluationLoop._evaluation_step(self, batch, batch_idx, dataloader_idx)
    372 self.batch_progress.increment_started()
    374 hook_name = "test_step" if trainer.testing else "validation_step"
--> 375 output = call._call_strategy_hook(trainer, hook_name, *step_kwargs.values())
    377 self.batch_progress.increment_processed()
    379 hook_name = "on_test_batch_end" if trainer.testing else "on_validation_batch_end"

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/trainer/call.py:291, in _call_strategy_hook(trainer, hook_name, *args, **kwargs)
    288     return None
    290 with trainer.profiler.profile(f"[Strategy]{trainer.strategy.__class__.__name__}.{hook_name}"):
--> 291     output = fn(*args, **kwargs)
    293 # restore current_fx when nested context
    294 pl_module._current_fx_name = prev_fx_name
```

```
File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/pytorch_lightning/strategies/strategy.py:379, in Strategy.validation_step(self, *args, **kwargs)
    377 with self.precision_plugin.val_step_context():
    378     assert isinstance(self.model, ValidationStep)
--> 379     return self.model.validation_step(*args, **kwargs)

File ~/cp_matgl/matgl/matgl/utils/training.py:59, in MatglLightningModuleMixin.validation_step(self, batch, batch_idx)
     52 def validation_step(self, batch: tuple, batch_idx: int):
     53     """Validation step.
     54
     55     Args:
     56         batch: Data batch.
     57         batch_idx: Batch index.
     58     """
---> 59     results, batch_size = self.step(batch)  # type: ignore
     60     self.log_dict(  # type: ignore
     61         {f"val_{key}": val for key, val in results.items()},
     62         batch_size=batch_size,
   (...)
     65         prog_bar=True,
     66     )
     67     return results["Total_Loss"]

File ~/cp_matgl/matgl/matgl/utils/training.py:334, in PotentialLightningModule.step(self, batch)
    332 torch.set_grad_enabled(True)
    333 g, l_g, state_attr, energies, forces, stresses = batch
--> 334 e, f, s, _ = self(g=g, state_attr=state_attr, l_g=l_g)
    335 f = f.to(torch.float)
    336 preds: tuple = (e, f, s)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File ~/cp_matgl/matgl/matgl/utils/training.py:322, in PotentialLightningModule.forward(self, g, l_g, state_attr)
    313 def forward(self, g: dgl.DGLGraph, l_g: dgl.DGLGraph | None = None, state_attr: torch.Tensor | None = None):
    314     """Args:
    315         g: dgl Graph
    316         l_g: Line graph
   (...)
    320         energy, force, stress, h
    321     """
```

```
--> 322     e, f, s, h = self.model(g=g, l_g=l_g, state_attr=state_attr)
    323     return e, f.float(), s, h

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File ~/cp_matgl/matgl/matgl/apps/pes.py:75, in Potential.forward(self, g, state_attr, l_g)
     73 if self.calc_forces:
     74     g.ndata["pos"].requires_grad_(True)
---> 75 total_energies = self.data_std * self.model(g=g, state_attr=state_attr, l_g=l_g) + self.data_mean
     76 if self.element_refs is not None:
     77     property_offset = torch.squeeze(self.element_refs(g))

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File ~/cp_matgl/matgl/matgl/models/_m3gnet.py:246, in M3GNet.forward(self, g, state_attr, l_g)
    243 g.edata["bond_vec"] = bond_vec.to(g.device)
    244 g.edata["bond_dist"] = bond_dist.to(g.device)
--> 246 expanded_dists = self.bond_expansion(g.edata["bond_dist"])
    247 if l_g is None:
    248     l_g = create_line_graph(g, self.threebody_cutoff)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/torch/nn/modules/module.py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self._forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
```

```
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File ~/cp_matgl/matgl/matgl/layers/_bond.py:65, in BondExpansion.forward(sel
f, bond_dist)
     56 def forward(self, bond_dist: torch.Tensor):
     57     """Forward.
     58
     59     Args:
   (...)
     63     bond_basis: Radial basis functions
     64     """
---> 65     bond_basis = self.rbf(bond_dist)
     66     return bond_basis

File ~/cp_matgl/matgl/matgl/layers/_basis.py:103, in SphericalBesselFunctio
n.__call__(self, r)
    101 if self.smooth:
    102     return self._call_smooth_sbf(r)
--> 103 return self._call_sbf(r)

File ~/cp_matgl/matgl/matgl/layers/_basis.py:121, in SphericalBesselFunctio
n._call_sbf(self, r)
    118     func = self.funcs[i]
    119     func_add1 = self.funcs[i + 1]
    120     results.append(
--> 121         func(r_c[:, None] * root[None, :] / self.cutoff) * factor /
torch.abs(func_add1(root[None, :]))
    122     )
    123 return torch.cat(results, axis=1)

File ~/miniconda3/envs/matgl/lib/python3.9/site-packages/torch/utils/_devic
e.py:62, in DeviceContext.__torch_function__(self, func, types, args, kwarg
s)
     60 if func in _device_constructors() and kwargs.get('device') is None:
     61     kwargs['device'] = self.device
---> 62 return func(*args, **kwargs)

RuntimeError: Expected all tensors to be on the same device, but found at le
ast two devices, cuda:0 and cpu!
```

In [ ]:
```
# This code just performs cleanup for this notebook.

for fn in ("dgl_graph.bin", "dgl_line_graph.bin", "state_attr.pt", "labels.j
    try:
        os.remove(fn)
    except FileNotFoundError:
        pass

shutil.rmtree("lightning_logs")
```

In [ ]:
```

```

In [ ]:
```

```