In [101]:
```
import matplotlib
matplotlib.use('Agg')

import datetime
import os
import sys
import json
import glob
import yaml
import math
import numpy as np
import re
from tqdm import tqdm

import matplotlib.pyplot as plt
%matplotlib inline

params = {
   'axes.labelsize': 14,
   'font.size': 14,
   'font.family': 'Roboto',
   'legend.fontsize': 20,
   'xtick.labelsize': 20,
   'ytick.labelsize': 20,
   'axes.labelsize': 25,
   'axes.titlesize': 25,
   'text.usetex': False,
   'figure.figsize': [12, 12]
}
matplotlib.rcParams.update(params)

import seaborn as sns
import pandas as pd

import ase
from ase.io.trajectory import Trajectory
import pickle
```

In [2]:
```
import torch
```

Visualizing model weights

In [3]:
```
root_dir = "/private/home/abhshkdz/projects/ocp-baselines"
checkpoint_path = os.path.join(root_dir, "checkpoints/2020-07-19-23-03-48-s
```

In [31]:
```
checkpoint = torch.load(checkpoint_path)
print(checkpoint.keys())
print(checkpoint['state_dict'].keys())
```

```
dict_keys(['epoch', 'state_dict', 'optimizer', 'normalizers', 'config'])
odict_keys(['atomic_mass', 'embedding.weight', 'distance_expansion.offse
t', 'interactions.0.mlp.0.weight', 'interactions.0.mlp.0.bias', 'interact
ions.0.mlp.2.weight', 'interactions.0.mlp.2.bias', 'interactions.0.conv.l
in1.weight', 'interactions.0.conv.lin2.weight', 'interactions.0.conv.lin
2.bias', 'interactions.0.conv.nn.0.weight', 'interactions.0.conv.nn.0.bia
s', 'interactions.0.conv.nn.2.weight', 'interactions.0.conv.nn.2.bias',
'interactions.0.lin.weight', 'interactions.0.lin.bias', 'interactions.1.m
lp.0.weight', 'interactions.1.mlp.0.bias', 'interactions.1.mlp.2.weight',
'interactions.1.mlp.2.bias', 'interactions.1.conv.lin1.weight', 'interact
ions.1.conv.lin2.weight', 'interactions.1.conv.lin2.bias', 'interactions.
1.conv.nn.0.weight', 'interactions.1.conv.nn.0.bias', 'interactions.1.con
v.nn.2.weight', 'interactions.1.conv.nn.2.bias', 'interactions.1.lin.weig
ht', 'interactions.1.lin.bias', 'interactions.2.mlp.0.weight', 'interacti
ons.2.mlp.0.bias', 'interactions.2.mlp.2.weight', 'interactions.2.mlp.2.b
ias', 'interactions.2.conv.lin1.weight', 'interactions.2.conv.lin2.weigh
t', 'interactions.2.conv.lin2.bias', 'interactions.2.conv.nn.0.weight',
'interactions.2.conv.nn.0.bias', 'interactions.2.conv.nn.2.weight', 'inte
ractions.2.conv.nn.2.bias', 'interactions.2.lin.weight', 'interactions.2.
lin.bias', 'lin1.weight', 'lin1.bias', 'lin2.weight', 'lin2.bias'])
```

In [35]:
```
checkpoint['epoch']
```

Out[35]:
```
50
```

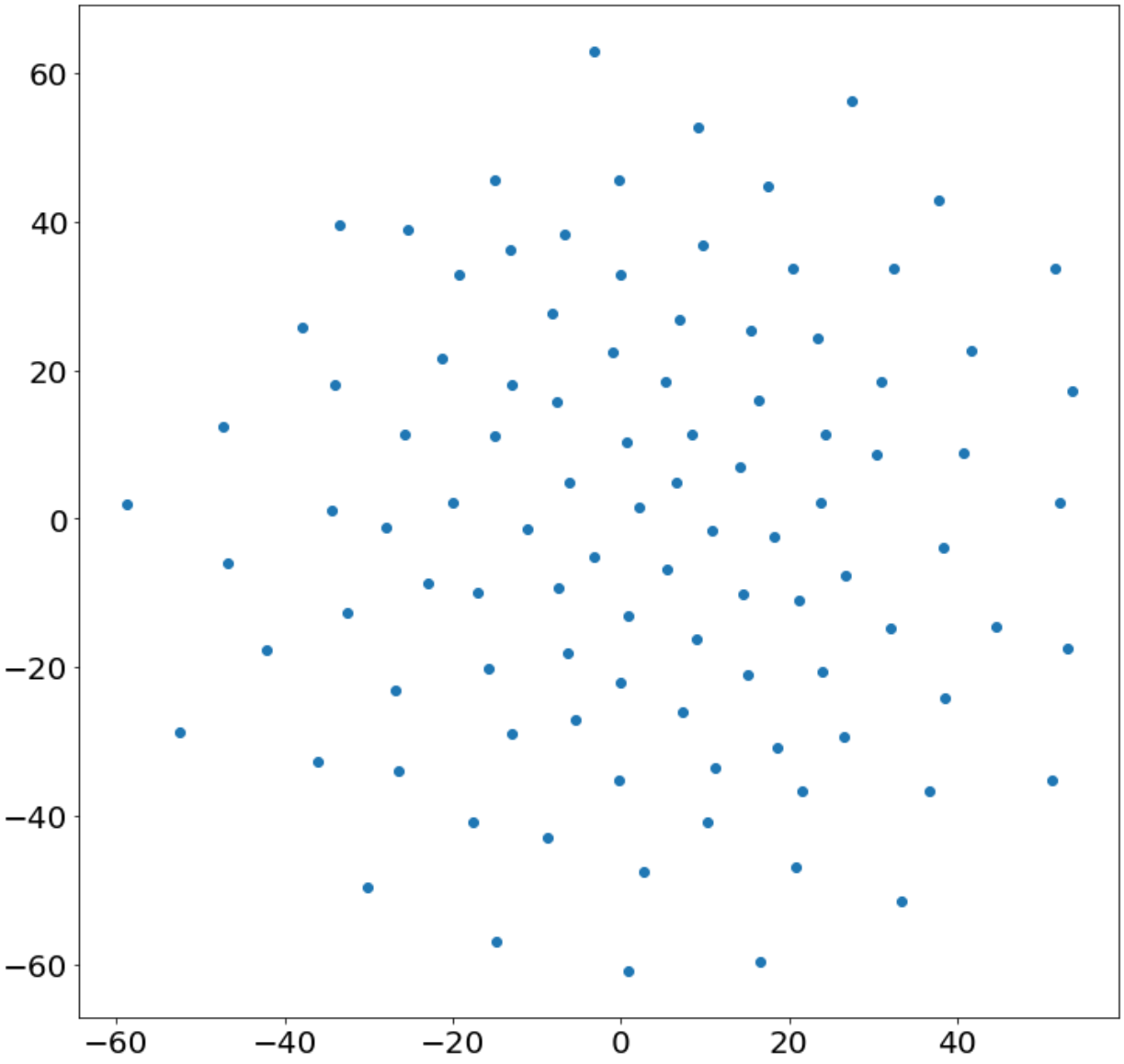
In [37]:
```
embedding = checkpoint['state_dict']['embedding.weight'].cpu()
embedding.shape
```

Out[37]:
```
torch.Size([100, 128])
```

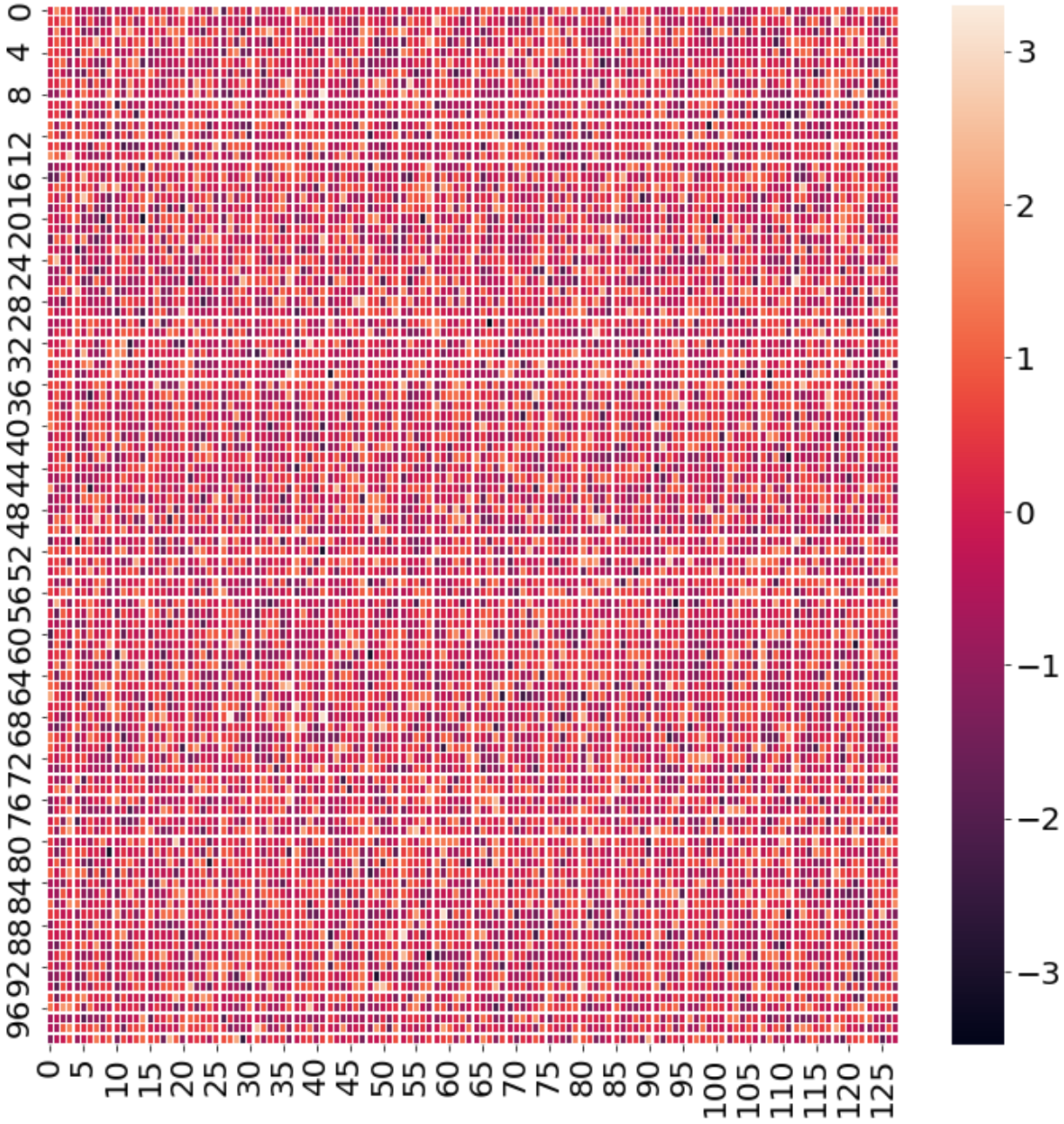
In [38]:
```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, random_state=0)
X_2d = tsne.fit_transform(embedding)
```

In [39]:
```
# color them eventually if you see any patterns
plt.scatter(X_2d[:, 0], X_2d[:, 1])
```

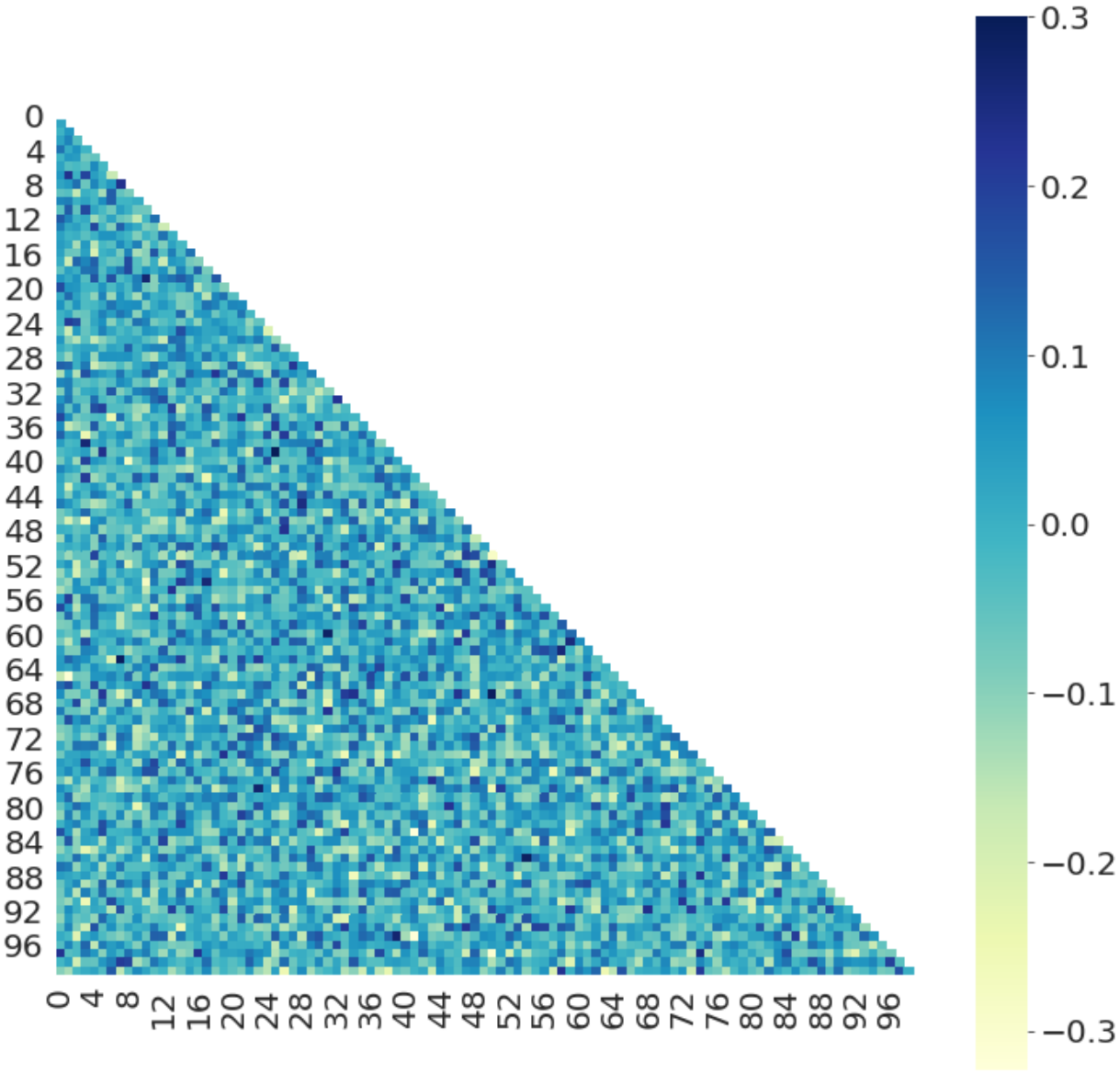
Out[39]: <matplotlib.collections.PathCollection at 0x7fcaa436ecf8>



In [40]:
```
ax = sns.heatmap(embedding, linewidth=0.2)
plt.show()
```



In [41]:
```
corr = np.corrcoef(embedding)
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True,  cmap="YlGnBu")
    plt.show()
```

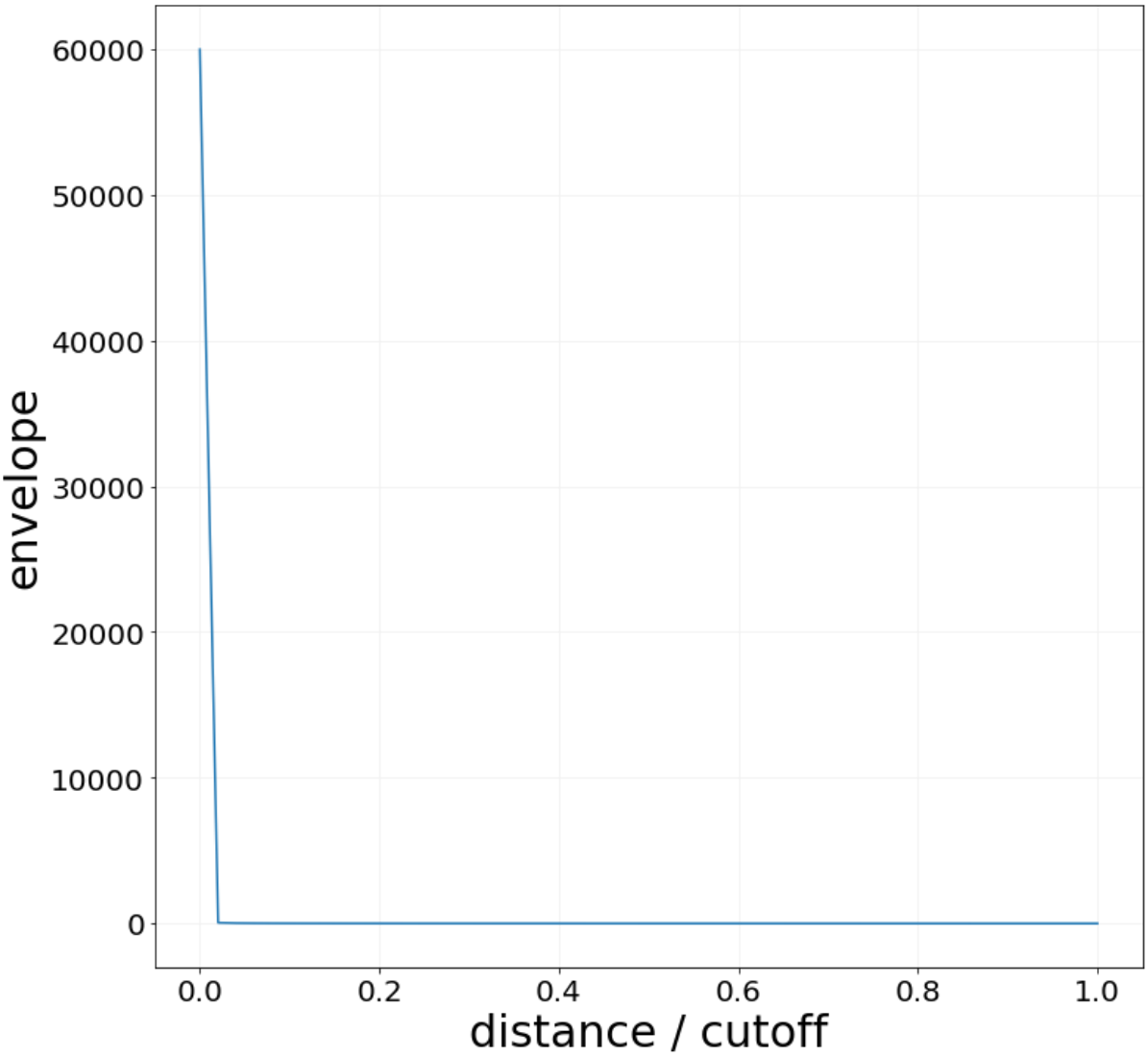


DimeNet

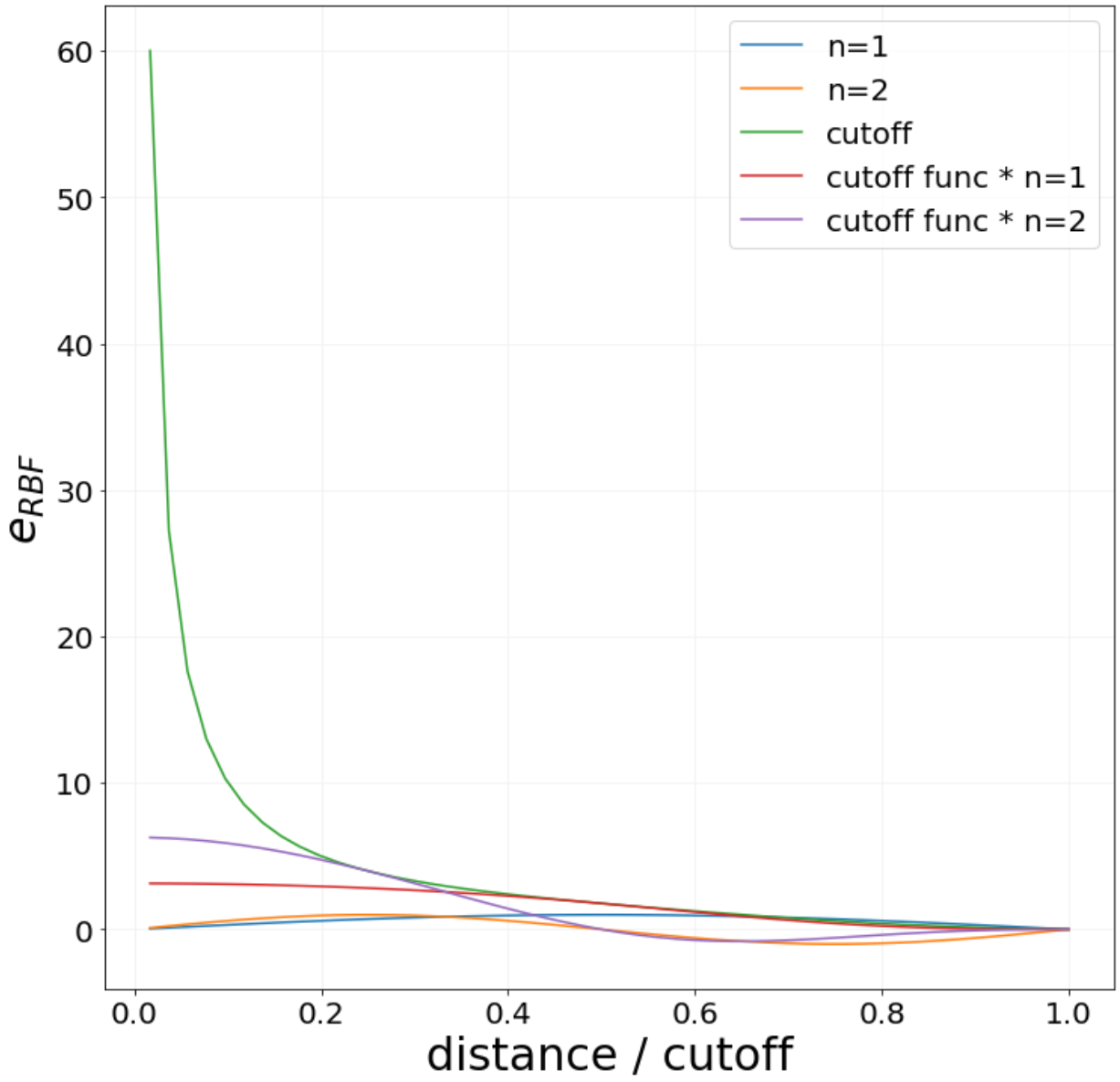
In [38]:
```
from math import sqrt, pi as PI


class Envelope(torch.nn.Module):
    def __init__(self, exponent):
        super(Envelope, self).__init__()
        self.p = exponent
        self.a = -(self.p + 1) * (self.p + 2) / 2
        self.b = self.p * (self.p + 2)
        self.c = -self.p * (self.p + 1) / 2

    def forward(self, x):
        p, a, b, c = self.p, self.a, self.b, self.c
        x_pow_p0 = x.pow(p)
        x_pow_p1 = x_pow_p0 * x
        return 1. / x + a * x_pow_p0 + b * x_pow_p1 + c * x_pow_p1 * x


class BesselBasisLayer(torch.nn.Module):
    def __init__(self, num_radial, cutoff=5.0, envelope_exponent=5):
        super(BesselBasisLayer, self).__init__()
        self.cutoff = cutoff
        self.envelope = Envelope(envelope_exponent)

        self.freq = torch.nn.Parameter(torch.Tensor(num_radial))

        self.reset_parameters()

    def reset_parameters(self):
        torch.arange(1, self.freq.numel() + 1, out=self.freq).mul_(PI)

    def forward(self, dist):
        dist = dist.unsqueeze(-1) / self.cutoff
        sine_out = (self.freq * dist).sin()
        return self.envelope(dist), (self.freq * dist).sin()
        # return self.envelope(dist) * (self.freq * dist).sin()
```

In [51]:
```
cutoff = 6.0
dist = torch.from_numpy(np.linspace(0.0001, cutoff, 50))
layer = Envelope(5)
out = layer(dist/cutoff)

print(out)

sns.lineplot(x=dist/cutoff, y=out)

plt.ylabel("envelope", fontsize=30)
plt.xlabel("distance / cutoff", fontsize=30)

plt.grid(color="0.95")
```

```
tensor([6.0000e+04, 4.8961e+01, 2.4490e+01, 1.6329e+01, 1.2248e+01, 9.798
4e+00,
        8.1652e+00, 6.9983e+00, 6.1227e+00, 5.4409e+00, 4.8946e+00, 4.446
4e+00,
        4.0714e+00, 3.7523e+00, 3.4766e+00, 3.2351e+00, 3.0209e+00, 2.828
7e+00,
        2.6541e+00, 2.4940e+00, 2.3456e+00, 2.2067e+00, 2.0756e+00, 1.950
9e+00,
        1.8315e+00, 1.7163e+00, 1.6048e+00, 1.4963e+00, 1.3906e+00, 1.287
4e+00,
        1.1865e+00, 1.0880e+00, 9.9191e-01, 8.9851e-01, 8.0801e-01, 7.207
5e-01,
        6.3708e-01, 5.5738e-01, 4.8206e-01, 4.1152e-01, 3.4612e-01, 2.861
6e-01,
        2.3189e-01, 1.8342e-01, 1.4075e-01, 1.0370e-01, 7.1878e-02, 4.465
5e-02,
        2.1113e-02, 0.0000e+00], dtype=torch.float64)
```


60000
50000
40000
30000
20000
10000
0
envelope
0.0
0.2
0.4
0.6
0.8
1.0
distance / cutoff

In [67]:
```
cutoff = 6.0
dist = torch.from_numpy(np.linspace(0.1, cutoff, 50))
layer = BesselBasisLayer(2, cutoff=6.0)
env_out, sine_out = layer(dist)

for i in range(sine_out.shape[1]):
    ax = sns.lineplot(x=dist/cutoff, y=sine_out[:, i], label="n={}".format(

sns.lineplot(x=dist/cutoff, y=env_out.squeeze(), label="cutoff")

for i in range(sine_out.shape[1]):
    ax = sns.lineplot(x=dist/cutoff, y=env_out.squeeze() * sine_out[:, i],

plt.ylabel("$e_{RBF}$", fontsize=30)
plt.xlabel("distance / cutoff", fontsize=30)

plt.grid(color="0.95")
```



Initial state to final energy prediction

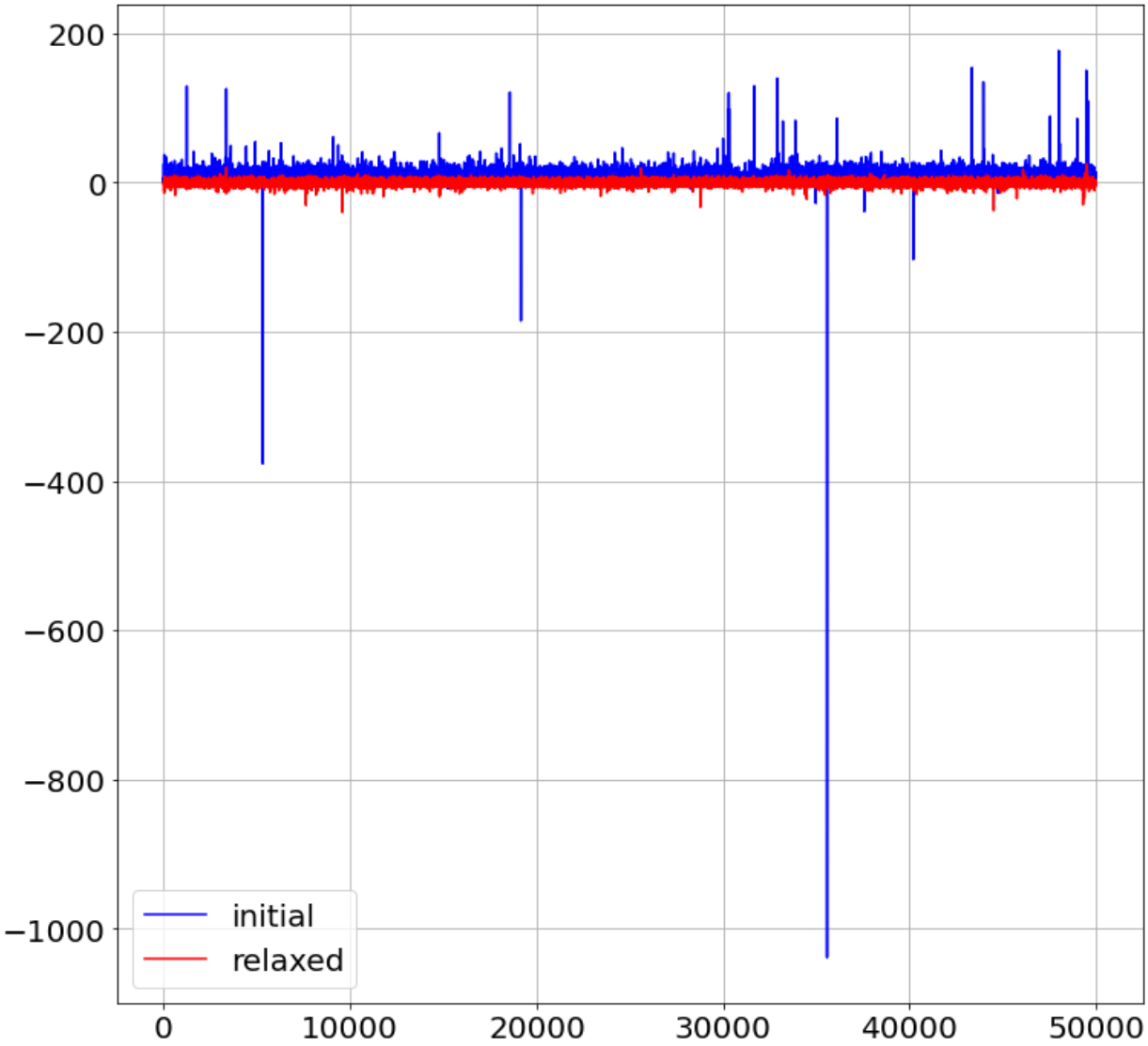
Dataset

In [117]:
```
from torch.utils.data import DataLoader
from ocpmodels.datasets import SinglePointLmdbDataset, data_list_collater

root_dir = "/private/home/abhshkdz/projects/ocp-baselines"

dataset_config = {
    "src": os.path.join(root_dir, "data/data/2020_07_25_ocp_is2re/50k/train
}
dataset = SinglePointLmdbDataset(dataset_config)

data_loader = DataLoader(
    dataset,
    batch_size=256,
    shuffle=False,
    collate_fn=data_list_collater,
    num_workers=16,
)

init_energies, relaxed_energies = [], []

for i, batch in tqdm(enumerate(data_loader)):
    init_energies.append(batch.y_init)
    relaxed_energies.append(batch.y_relaxed)

init_energies = torch.cat(init_energies, 0).view(-1, 1)
relaxed_energies = torch.cat(relaxed_energies, 0).view(-1, 1)
```

```
196it [00:05, 35.83it/s]
```

In [118]:
```
plt.plot(init_energies, color="blue", label="initial")
plt.plot(relaxed_energies, color="red", label="relaxed")
plt.grid(0.8)
plt.legend()
```

Out[118]: <matplotlib.legend.Legend at 0x7f47ed76b3c8>



In [120]:
```
init_outliers = (init_energies < -100).nonzero()[:, 0].tolist()
relaxed_outliers = (relaxed_energies < -100).nonzero()[:, 0].tolist()

print(init_outliers, relaxed_outliers)
```

```
[5312, 19154, 35574, 40194] []
```

In [121]:
```
adslab_ref = pickle.load(open("/checkpoint/mshuaibi/mappings/adslab_ref_ene
with open(os.path.join("/checkpoint/mshuaibi/ocpdata_reset_07_13_20/ocpdata
    raw_traj_files = f.read().splitlines()
```

In [135]:
```
for i in init_outliers:
    print(init_energies[i].item())

    fpath = raw_traj_files[i+5]
    randomid = fpath.split("/")[4].split(".")[0]

    print(fpath)
    traj = Trajectory(fpath)

    energy = traj[0].get_potential_energy(apply_constraint=False)
    ref = adslab_ref[randomid]

    print(energy)
    print(ref)
    print(energy-ref, "\n")
```

```
-376.7870788574219
/checkpoint/sidgoyal/electro_done/random1015717.traj
-327.91421888
-332.5292841
4.615065219999963

-185.327392578125
/checkpoint/sidgoyal/electro_done/random1050241.traj
-280.84412423
-286.73788265999997
5.893758429999991

-1038.3223876953125
/checkpoint/sidgoyal/electro_done/random1248557.traj
-391.53414044
-393.97503852999995
2.440898089999962

-103.18209838867188
/checkpoint/sidgoyal/electro_done/random1256180.traj
-736.01101592
-632.82891803
-103.18209789000002
```

In [103]:
```
for i in relaxed_outliers:
    print(relaxed_energies[i].item())

    fpath = raw_traj_files[i+1]
    randomid = fpath.split("/")[4].split(".")[0]

    print(fpath)
    traj = Trajectory(fpath)

    energy = traj[-1].get_potential_energy(apply_constraint=False)
    ref = adslab_ref[randomid]

    print(energy)
    print(ref)
    print(energy-ref, "\n")
```

```
-955.1636962890625
/checkpoint/sidgoyal/electro_done/random1031649.traj
-1715.99962964
-760.83591632
-955.1637133199999

-11676.8515625
/checkpoint/sidgoyal/electro_done/random1255319.traj
-11983.41803287
-306.56609184
-11676.85194103
```

In [114]:
```
print(relaxed_energies[(relaxed_energies > -50)].mean())
print(relaxed_energies[(relaxed_energies > -50)].std())

print(relaxed_energies.mean())
print(relaxed_energies.std())
```

```
tensor(-1.3223)
tensor(2.5169)
tensor(-1.5749)
tensor(52.4494)
```

Hyperparam sweeps

In [7]:
```
root_dir = "/private/home/abhshkdz/projects/ocp-baselines"

# schnet.
# exp_log = ["2020-08-04-02-26-59PM.log", "2020-08-04-10-51-52AM.log", "202
# cgcnn
# exp_log = ["2020-08-04-11-23-42PM.log"]

# schnet after pbc fix.
# 10k
# exp_log = ["2020-08-05-03-29-40PM.log"]
# 100k
# exp_log = ["2020-08-05-06-06-15PM.log", "2020-08-05-06-05-47PM.log"]

# cgcnn after pbc fix.
# 10k
# exp_log = ["2020-08-05-03-31-34PM.log"]
# 100k
# exp_log += ["2020-08-05-06-10-44PM.log"]
# 300k
# exp_log += ["2020-08-06-12-38-42AM.log"]
# all
# exp_log = ["2020-08-07-12-20-06AM.log"]

# CO 5.9k
# exp_log = ["2020-08-06-12-12-13AM.log"]
# CO 5.9k, RS2RE
# exp_log = ["2020-08-06-12-24-12AM.log"]

# Another pbc fix.
# train: OCP CO 5.9k, val: OCP CO 0.6k
# exp_log = ["2020-08-12-10-24-29PM.log"]

# train: UlissigroupCO 15.8k, val: UlissigroupCO 1k
# exp_log = ["2020-08-12-11-42-06PM.log"]

# train: UlissigroupCO 15.8k + OCP CO 5.9k, val: OCP CO 0.6k
# exp_log = ["2020-08-13-08-43-31AM.log"]

# train: UlissigroupCO 15.8k + OCP CO 5.9k, val: UlissigroupCO 1k
# exp_log = ["2020-08-13-08-43-36AM.log"]



# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=12
# exp_log = ["2020-08-18-06-33-38PM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=30
# exp_log = ["2020-08-19-06-03-57PM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=50
# exp_log = ["2020-08-19-06-04-25PM.log", "2020-08-19-06-42-59PM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=100
# exp_log = ["2020-08-19-06-04-43PM.log", "2020-08-19-06-43-37PM.log", "202

# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=200
```

```
# exp_log = ["2020-08-19-06-05-09PM.log", "2020-08-19-06-44-11PM.log", "202

# train: OCP CO 7.7k, val: OCP CO 0.6k, DimeNet, max_nbr=200
# exp_log = ["2020-08-20-03-12-15PM.log"]
# exp_log = ["2020-08-20-04-42-22PM.log"]


# train: OCP CO 7.7k, val: OCP CO 0.6k, CGCNN, max_nbr=12
# exp_log = ["2020-08-13-08-27-26PM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, CGCNN, max_nbr=30
# exp_log = ["2020-08-20-12-50-44AM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, CGCNN, max_nbr=50
# exp_log = ["2020-08-20-12-51-18AM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, CGCNN, max_nbr=100
# exp_log = ["2020-08-20-12-51-37AM.log"]

# train: OCP CO 7.7k, val: OCP CO 0.6k, CGCNN, max_nbr=200
# exp_log = ["2020-08-20-12-52-02AM.log"]


# OCP final splits.
# CGCNN, 1k
# exp_log = ["2020-08-24-12-52-52AM.log"]

# CGCNN, 10k
# exp_log = ["2020-08-24-01-01-17AM.log"]

# CGCNN, 100k
# exp_log = ["2020-08-24-01-08-35AM.log"]

# CGCNN, All
# exp_log = ["2020-08-24-04-14-20PM.log"]

# DimeNet, 1k
# exp_log = ["2020-08-25-01-42-28AM.log"]
# exp_log = ["2020-08-25-01-42-28AM.log", "2020-08-26-11-56-27PM.log", "202

# DimeNet, 10k
# exp_log = ["2020-08-25-01-43-21AM.log"]
# exp_log = ["2020-08-26-11-57-00PM.log"]

# DimeNet, 100k
# exp_log = ["2020-08-26-11-57-23PM.log"]

# DimeNet, All
# exp_log = ["2020-08-26-02-37-43AM.log"]
# exp_log = ["2020-08-27-10-18-19PM.log"]

# SchNet, 1k
# exp_log = ["2020-08-28-12-21-14AM.log"]
exp_log = ["2020-09-01-01-04-56AM.log"]

# SchNet, 10k
# exp_log = ["2020-08-28-12-24-09AM.log"]
```

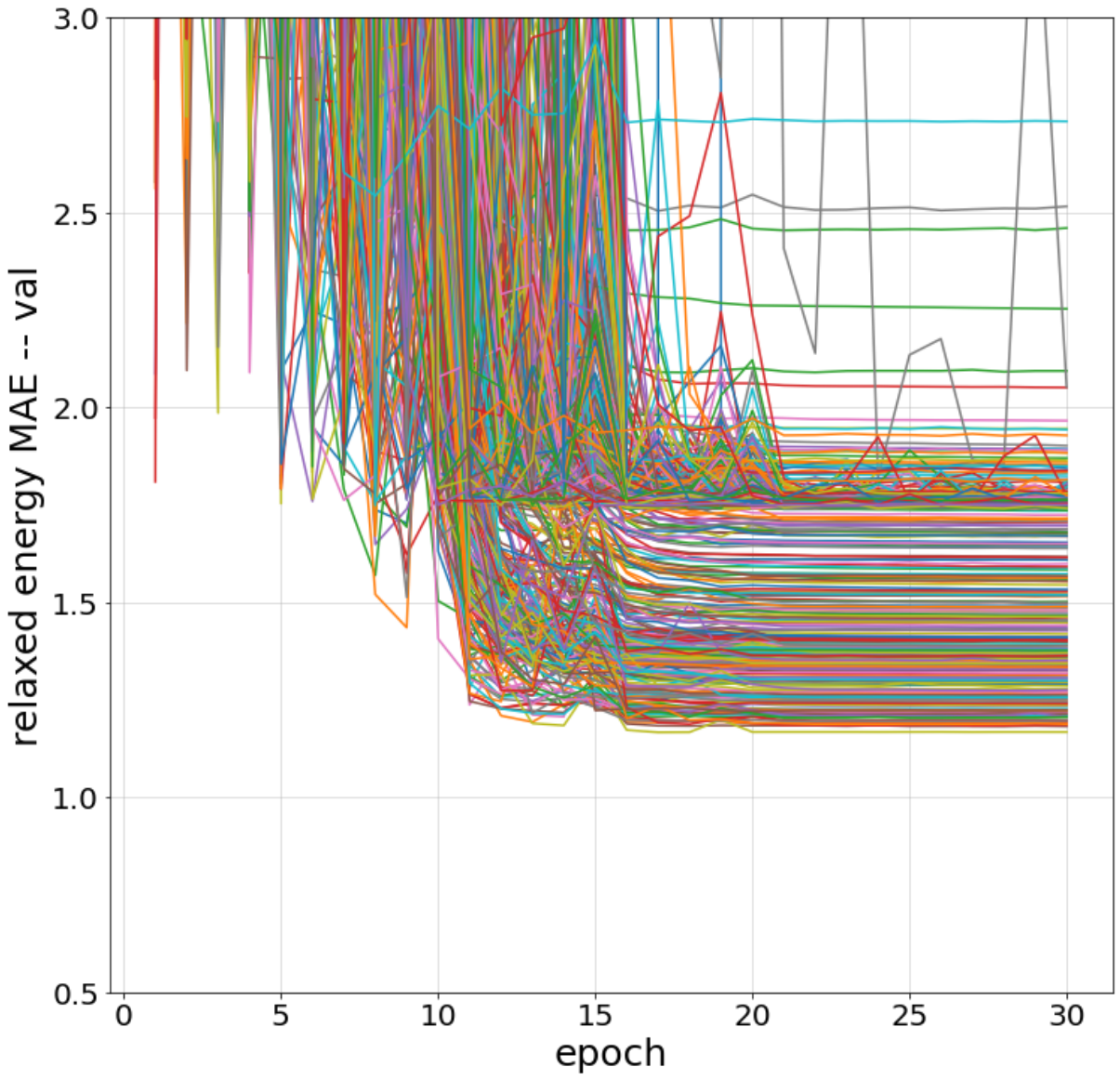
```
# SchNet, 100k
# exp_log = ["2020-08-28-12-25-39AM.log"]

# SchNet, All
# exp_log = ["2020-08-28-12-27-04AM.log"]

exps = []
for e in exp_log:
    with open(os.path.join(root_dir, "logs/slurm/exp", e), "r") as f:
        exps += f.read().splitlines()

for i in range(len(exps)):
    exps[i] = json.loads(exps[i])

print(exps[0])
```

{'config': {'model': {'name': 'schnet', 'hidden_channels': 64, 'num_filte
rs': 64, 'num_interactions': 1, 'num_gaussians': 100, 'cutoff': 6.0, 'use
_pbc': True, 'regress_forces': False}, 'optim': {'batch_size': 16, 'eval_
batch_size': 64, 'num_workers': 32, 'lr_initial': 0.05, 'lr_gamma': 0.1,
'lr_milestones': [10, 15, 20], 'warmup_epochs': 3, 'warmup_factor': 0.2,
'max_epochs': 30, 'num_gpus': 1}, 'trainer': 'energy', 'dataset': [{'sr
c': 'data/data/2020_08_19_ocp_is2re/1k/train/data.lmdb', 'normalize_label
s': True, 'target_mean': -1.4003570079803467, 'target_std': 2.24838471412
6587}, {'src': 'data/data/2020_08_19_ocp_is2re/all/val_is/data.lmdb'}],
'logger': 'tensorboard', 'task': {'dataset': 'single_point_lmdb', 'descri
ption': 'Relaxed state energy prediction from initial structure.', 'typ
e': 'regression', 'metric': 'mae', 'labels': ['relaxed energy']}, 'identi
fier': 'schnet_ocp_is2re_1k_run0', 'seed': 1, 'is_debug': False, 'is_vi
s': False, 'print_every': 10, 'submit': True, 'local_rank': 0, 'distribut
ed_port': 13356, 'world_size': 1, 'distributed_backend': 'nccl'}, 'slurm_
id': '29700237_0', 'timestamp': '01:04:56AM PDT Sep 01, 2020'}

In [8]:
```
def read_ocp_log(slurm_job_id):
    try:
        log_path = glob.glob(root_dir + "/logs/slurm/{}/*.out".format(slurm
    except:
        return [], []
    log = open(log_path, "r").read().splitlines()
    train, val = [], []
    is_next_val = False
    for l in log:
        fragments = l.split(",")
        if "ERROR" in fragments[0]:
            return train, val
        # train.
        elif is_next_val is False and fragments[0].startswith("epoch"):
            metrics = {}
            for f in fragments:
                sub = f.split(":")
                metrics[sub[0].strip()] = float(sub[1].strip())
            train.append(metrics)
        # val.
        elif is_next_val is True:
            epoch = round(metrics["epoch"])
            metrics = {}
            try:
                for f in fragments:
                    sub = f.split(":")
                    metrics[sub[0].strip()] = float(sub[1].strip())
            except:
                return train, val
                print(slurm_job_id)
            metrics["epoch"] = epoch
            val.append(metrics)
            is_next_val = False
        if l == "### Evaluating on val.":
            is_next_val = True
    return train, val

logs = [read_ocp_log(e["slurm_id"]) for e in exps]
```

In [9]:
```
exps_idx = list(range(len(exps)))

# search_str = "use_pbcTrue"
# exps_idx = [i for i in exps_idx if search_str in exps[i]["config"]["ident

for i in exps_idx:
    plt.plot([e["epoch"] for e in logs[i][1]],
             [e["relaxed energy/mae"] for e in logs[i][1]],
             label=exps[i]["config"]["identifier"])

plt.ylim(0.5, 3.0)
plt.xlabel("epoch")
plt.ylabel("relaxed energy MAE -- val")
plt.grid(alpha=0.5)

# plt.savefig(os.path.join(root_dir, "notebooks/figures", exp_log+".energy_
#             dpi=150,
#             bbox_inches="tight")
```

```
findfont: Font family ['Roboto'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Roboto'] not found. Falling back to DejaVu Sans.
```



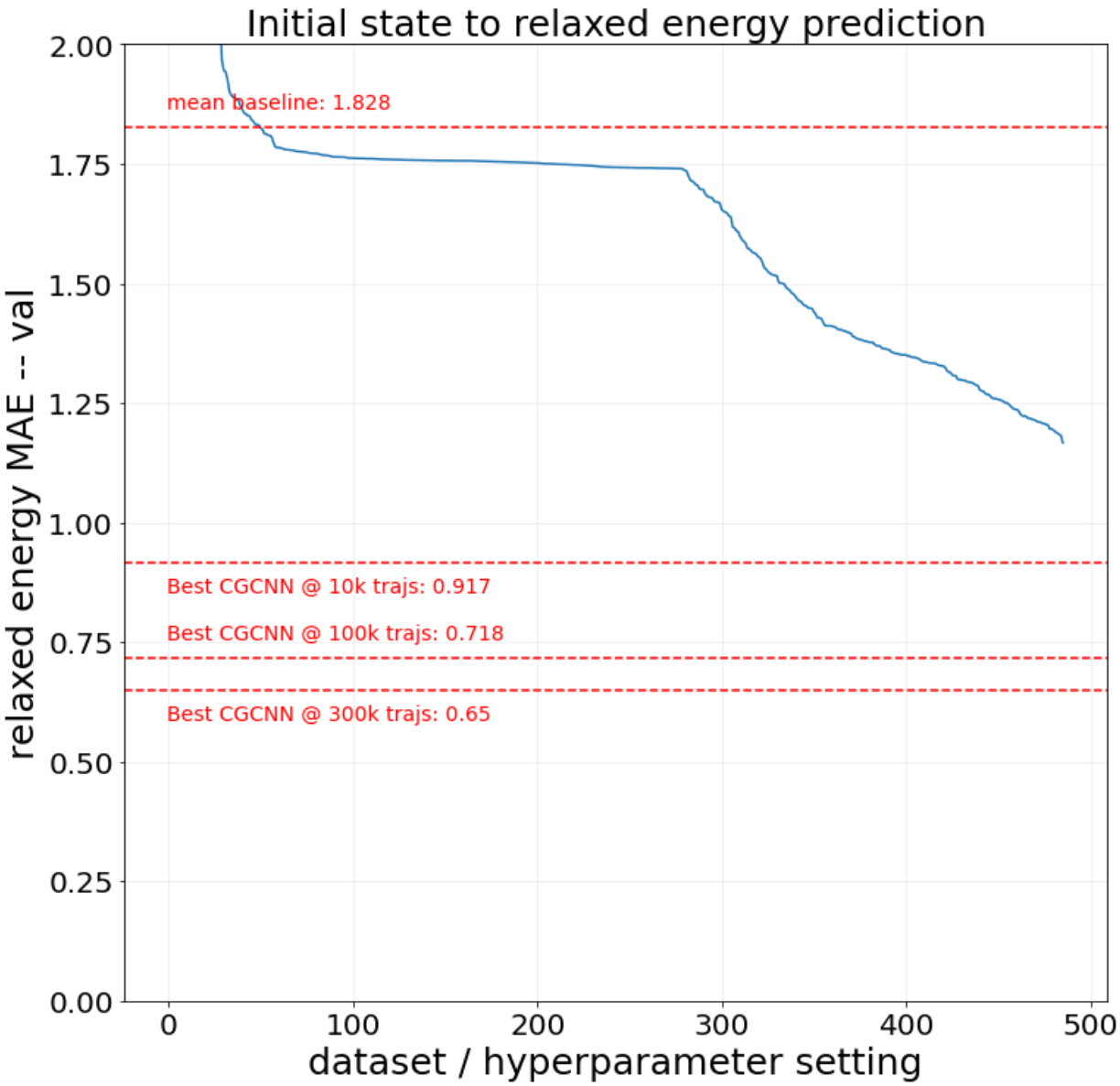
In [10]:
```
best_val_maes = [(i, min([m["relaxed energy/mae"] for m in logs[i][1]])) fo
print("Jobs completed:", len(best_val_maes))

best_val_maes_sorted = sorted(best_val_maes, key=lambda x: x[1])[::-1]
configs_sorted = [(i[1], exps[i[0]]) for i in best_val_maes_sorted][::-1]

print("Best 5:")
for i in range(5):
    print(configs_sorted[i][0], configs_sorted[i][1]["slurm_id"], configs_s

plt.plot(list(range(len(best_val_maes))), [i[1] for i in best_val_maes_sort

plt.grid(alpha=0.2)
plt.ylim(0, 2.0)
plt.xlabel("dataset / hyperparameter setting")
plt.ylabel("relaxed energy MAE -- val")
plt.title("Initial state to relaxed energy prediction")

x_text = -1

# baselines / best at dataset size.
mean_baseline = 1.828
plt.axhline(y=mean_baseline, color='r', linestyle='--')
plt.text(x_text, mean_baseline+0.05, "mean baseline: {}".format(mean_baseli

# best_at_10k = 0.956
# plt.axhline(y=best_at_10k, color='r', linestyle='--')
# plt.text(x_text, best_at_10k+0.05, "Best SchNet @ 10k trajs: {}".format(b

best_at_10k = 0.917
plt.axhline(y=best_at_10k, color='r', linestyle='--')
plt.text(x_text, best_at_10k-0.05, "Best CGCNN @ 10k trajs: {}".format(best

# best_at_50k = 0.798
# plt.axhline(y=best_at_50k, color='r', linestyle='--')
# plt.text(x_text, best_at_50k+0.05, "Best SchNet @ 50k trajs: {}".format(b

# best_at_100k = 0.736
# plt.axhline(y=best_at_100k, color='r', linestyle='--')
# plt.text(x_text, best_at_100k+0.05, "Best SchNet @ 100k trajs: {}".format

best_at_100k = 0.718
plt.axhline(y=best_at_100k, color='r', linestyle='--')
plt.text(x_text, best_at_100k+0.05, "Best CGCNN @ 100k trajs: {}".format(be

best_at_100k = 0.650
plt.axhline(y=best_at_100k, color='r', linestyle='--')
plt.text(x_text, best_at_100k-0.05, "Best CGCNN @ 300k trajs: {}".format(be

# plt.savefig(os.path.join(root_dir, "notebooks/figures", exp_log+".hparam.
# plt.savefig(os.path.join(root_dir, "notebooks/figures", "overall.hparam.p
#             dpi=150,
#             bbox_inches="tight")
```

```
Jobs completed: 486
```

```
Best 5:
1.167 29700237_148 01:04:56AM PDT Sep 01, 2020 schnet_ocp_is2re_1k_run1
48
1.1823 29700237_145 01:04:56AM PDT Sep 01, 2020 schnet_ocp_is2re_1k_run
145
1.1839 29700237_353 01:04:56AM PDT Sep 01, 2020 schnet_ocp_is2re_1k_run
353
1.1877 29700237_371 01:04:56AM PDT Sep 01, 2020 schnet_ocp_is2re_1k_run
371
1.189 29700237_101 01:04:56AM PDT Sep 01, 2020 schnet_ocp_is2re_1k_run1
01
```

Out[10]: `Text(-1, 0.6, 'Best CGCNN @ 300k trajs: 0.65')`

```
findfont: Font family ['Roboto'] not found. Falling back to DejaVu Sans.
```



Hyperparameter sweeps -- S2EF

In [ ]:
```
exp_log = ["2020-09-02-04-10-47AM.log"]
```

derivatives with cell offsets

In [62]:
```
r = torch.randn(3).requires_grad_(True)
offset = torch.randn(3)

model = torch.nn.Linear(3, 1)
optimizer = torch.optim.SGD(model.parameters(), 0.001)

# with offsets.
optimizer.zero_grad()
out1 = model(r + offset)
dout1 = torch.autograd.grad(
    out1,
    r,
    grad_outputs=torch.ones_like(out1),
    create_graph=True,
)[0]
print(dout1)

# without offsets.
optimizer.zero_grad()
out2 = model(r + offset)
dout2 = torch.autograd.grad(
    out1,
    r,
    grad_outputs=torch.ones_like(out2),
    create_graph=True,
)[0]
print(dout2)
```

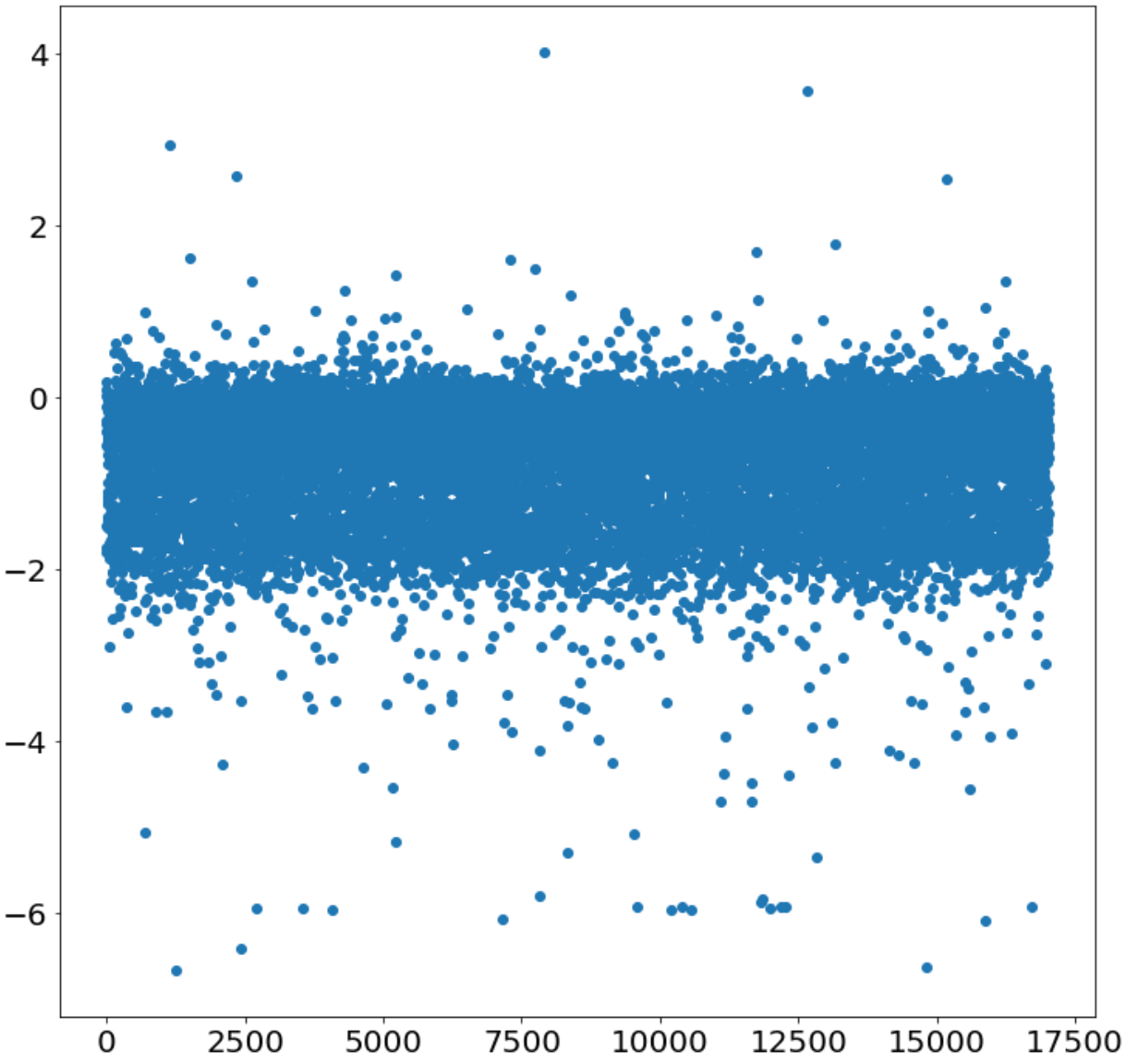
```
tensor([ 0.1242, -0.3024, -0.5177], grad_fn=<SqueezeBackward1>)
tensor([ 0.1242, -0.3024, -0.5177], grad_fn=<SqueezeBackward1>)
```

UlissigroupCO + IS2RE

In [12]:
```
root_dir = "/private/home/abhshkdz/projects/ocp-baselines"
uco_init_data = torch.load(os.path.join(root_dir,
                                        "data/data/2020_02_16_ulissigroup_c
```

In [20]:
```
plt.scatter(list(range(uco_init_data[0].y.shape[0])), uco_init_data[0].y)
```

Out[20]: <matplotlib.collections.PathCollection at 0x7f2d6c381be0>

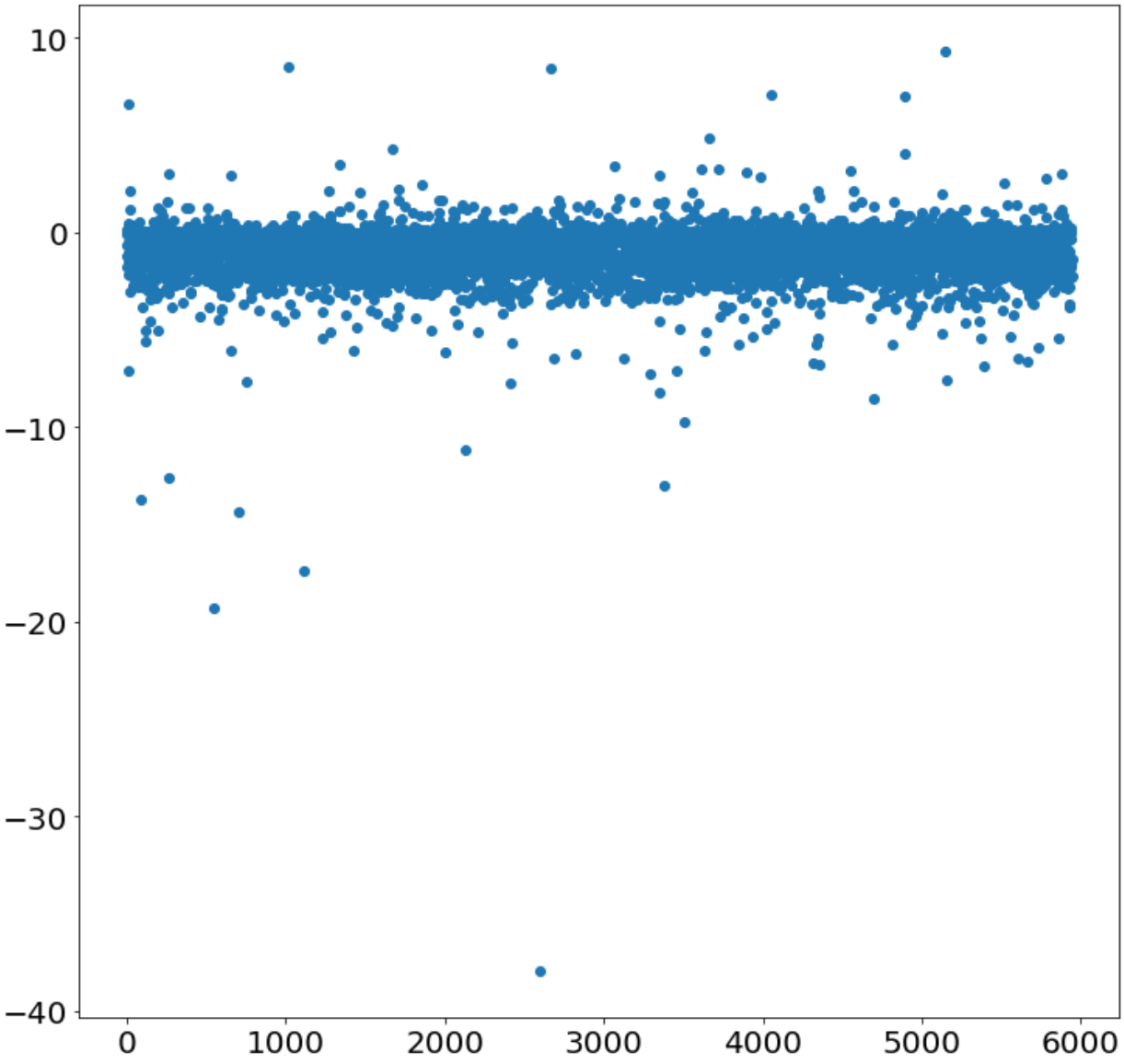


In [209]:
```
from torch.utils.data import DataLoader
from ocpmodels.datasets import SinglePointLmdbDataset, data_list_collater

dataset_config = {
    "src": "../data/data/2020_08_05_ocp_is2re_co/train/data.lmdb",
}

dataset = SinglePointLmdbDataset(dataset_config)
data_loader = DataLoader(
    dataset,
    batch_size=256,
    shuffle=False,
    collate_fn=data_list_collater,
    num_workers=16,
)

energies = []

for i, batch in tqdm(enumerate(data_loader)):
    energies.append(batch.y_relaxed)

energies = torch.cat(energies, 0).view(-1, 1)
```

```
24it [00:02, 10.99it/s]
```

In [210]:
```
plt.scatter(list(range(len(energies))), energies)
```

Out[210]:
```
<matplotlib.collections.PathCollection at 0x7f2ced37fe10>
```

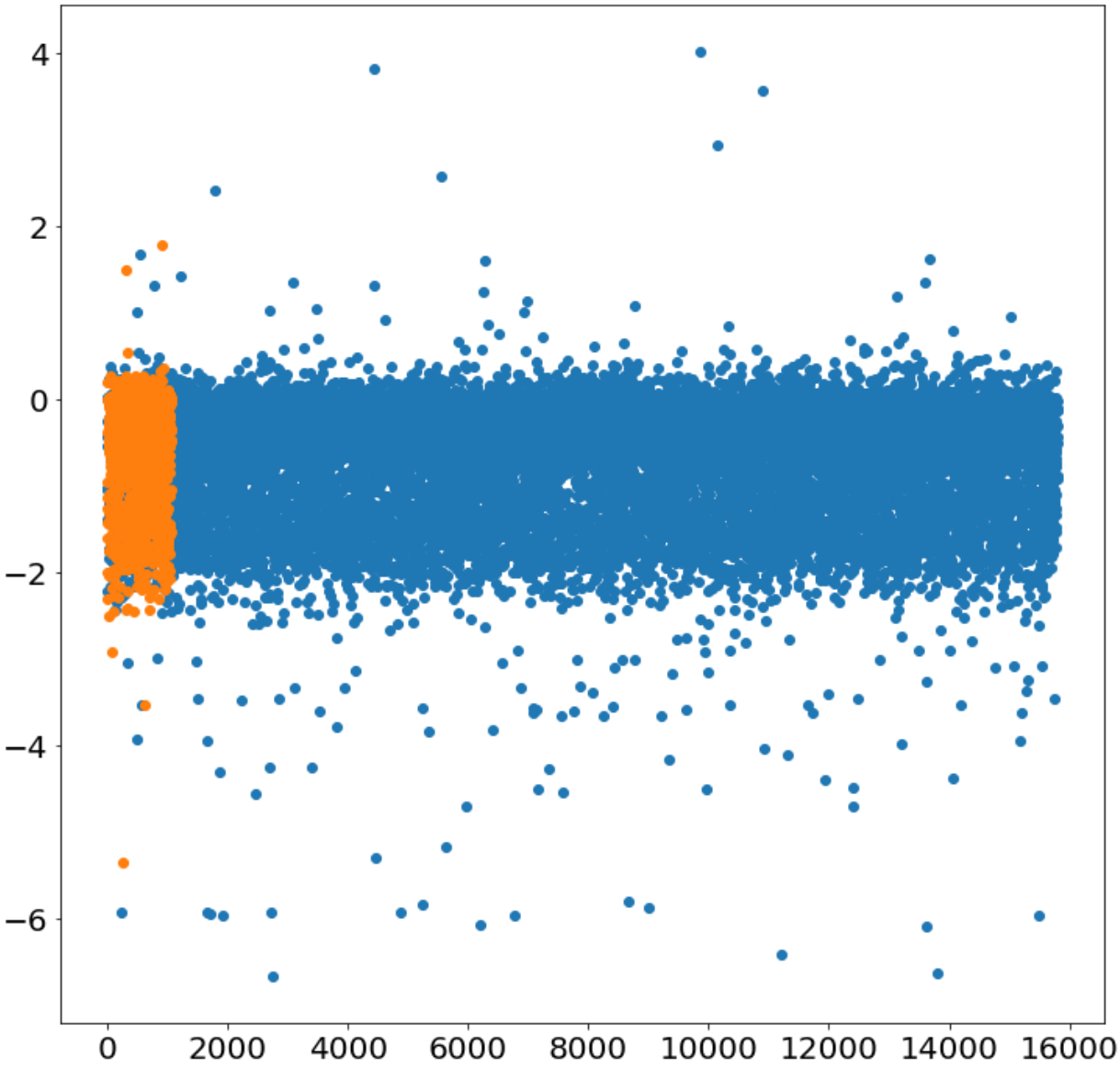


In [289]:
```
from torch.utils.data import DataLoader
from ocpmodels.datasets import SinglePointLmdbDataset, data_list_collater

srcs = ["../data/data/2020_08_11_ulissigroup_co_trajs/train/data.lmdb",
        "../data/data/2020_08_11_ulissigroup_co_trajs/val/data.lmdb"]

# srcs = ["../data/data/2020_08_05_ocp_is2re_co/train/data.lmdb",
#         "../data/data/2020_08_05_ocp_is2re_co/val/data.lmdb"]

dataset_config = {
    "src": srcs[0],
}
dataset = SinglePointLmdbDataset(dataset_config)
data_loader = DataLoader(
    dataset,
    batch_size=256,
    shuffle=False,
    collate_fn=data_list_collater,
    num_workers=16,
)
energies1 = []
for i, batch in tqdm(enumerate(data_loader)):
    energies1.append(batch.y_relaxed)
energies1 = torch.cat(energies1, 0).view(-1, 1)


dataset_config = {
    "src": srcs[1],
}
dataset = SinglePointLmdbDataset(dataset_config)
data_loader = DataLoader(
    dataset,
    batch_size=256,
    shuffle=False,
    collate_fn=data_list_collater,
    num_workers=16,
)
energies2 = []
for i, batch in tqdm(enumerate(data_loader)):
    energies2.append(batch.y_relaxed)
energies2 = torch.cat(energies2, 0).view(-1, 1)
```

```
62it [00:03, 20.19it/s]
5it [00:01,  3.10it/s]
```

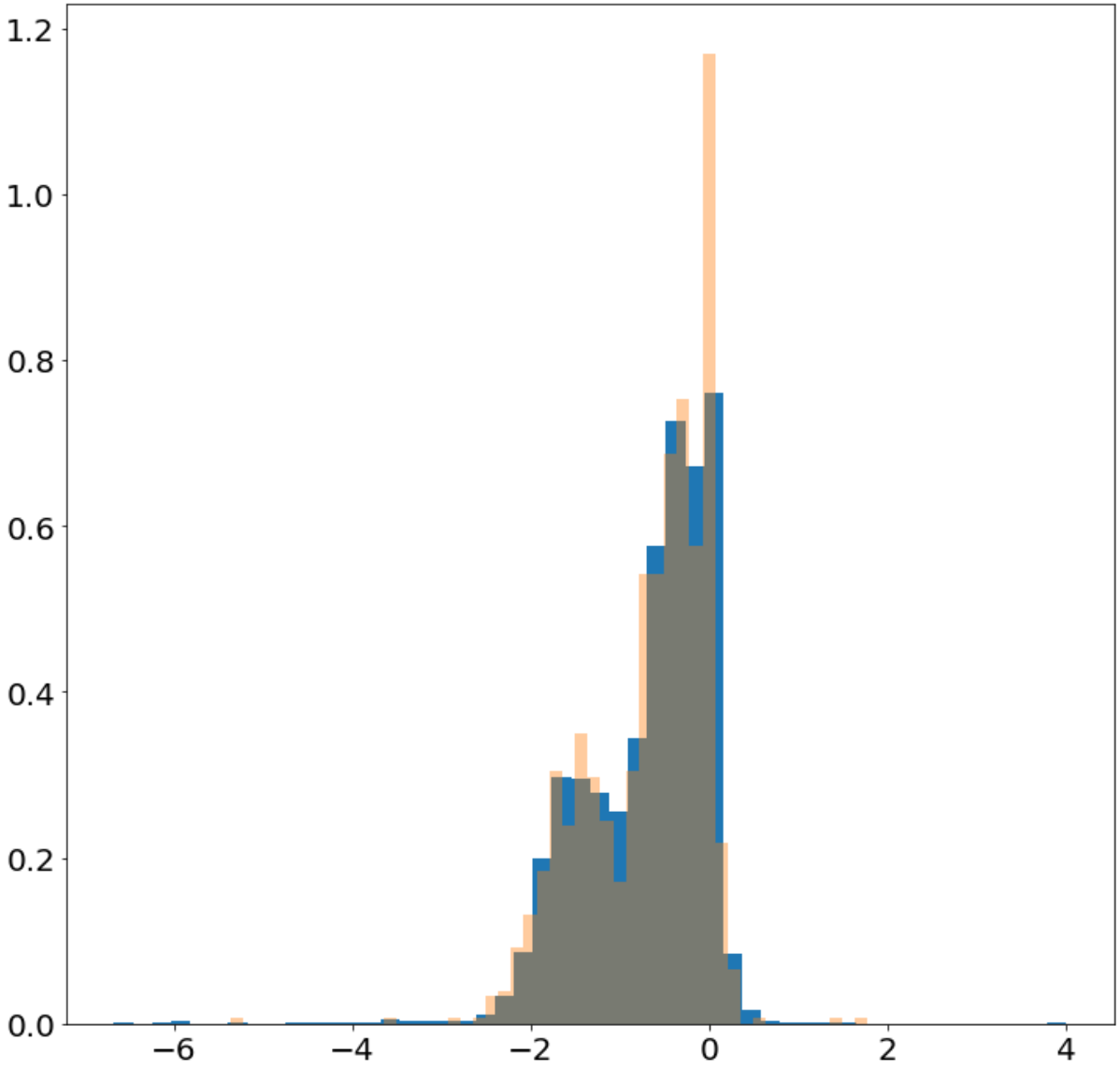
In [290]:
```
plt.scatter(list(range(len(energies1))), energies1)
plt.scatter(list(range(len(energies2))), energies2)
```

Out[290]: <matplotlib.collections.PathCollection at 0x7f2c2f387860>

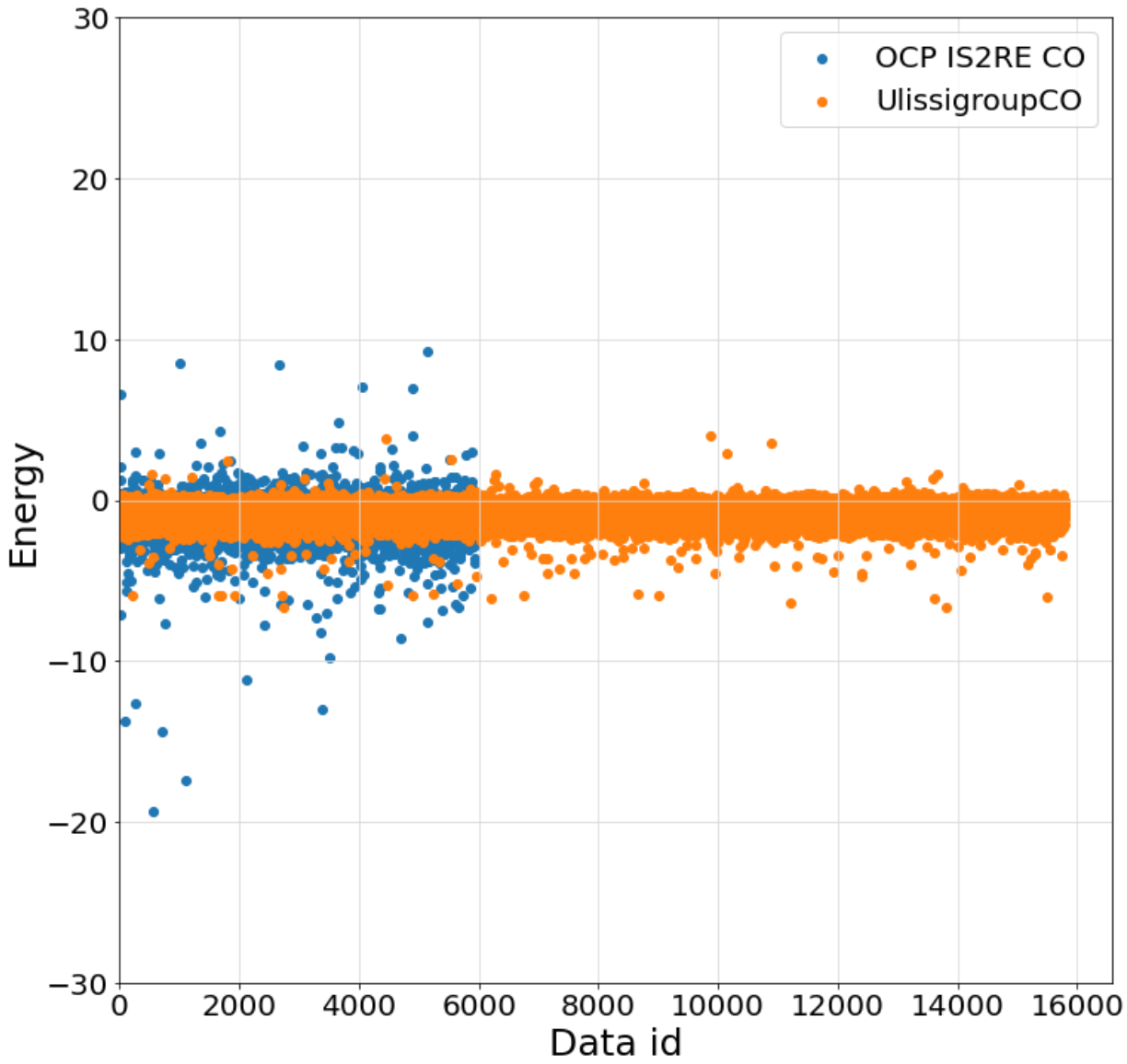


In [291]:
```
plt.hist(energies1.flatten(), bins=50, density=True)
plt.hist(energies2.flatten(), bins=50, density=True, alpha=0.4)
```

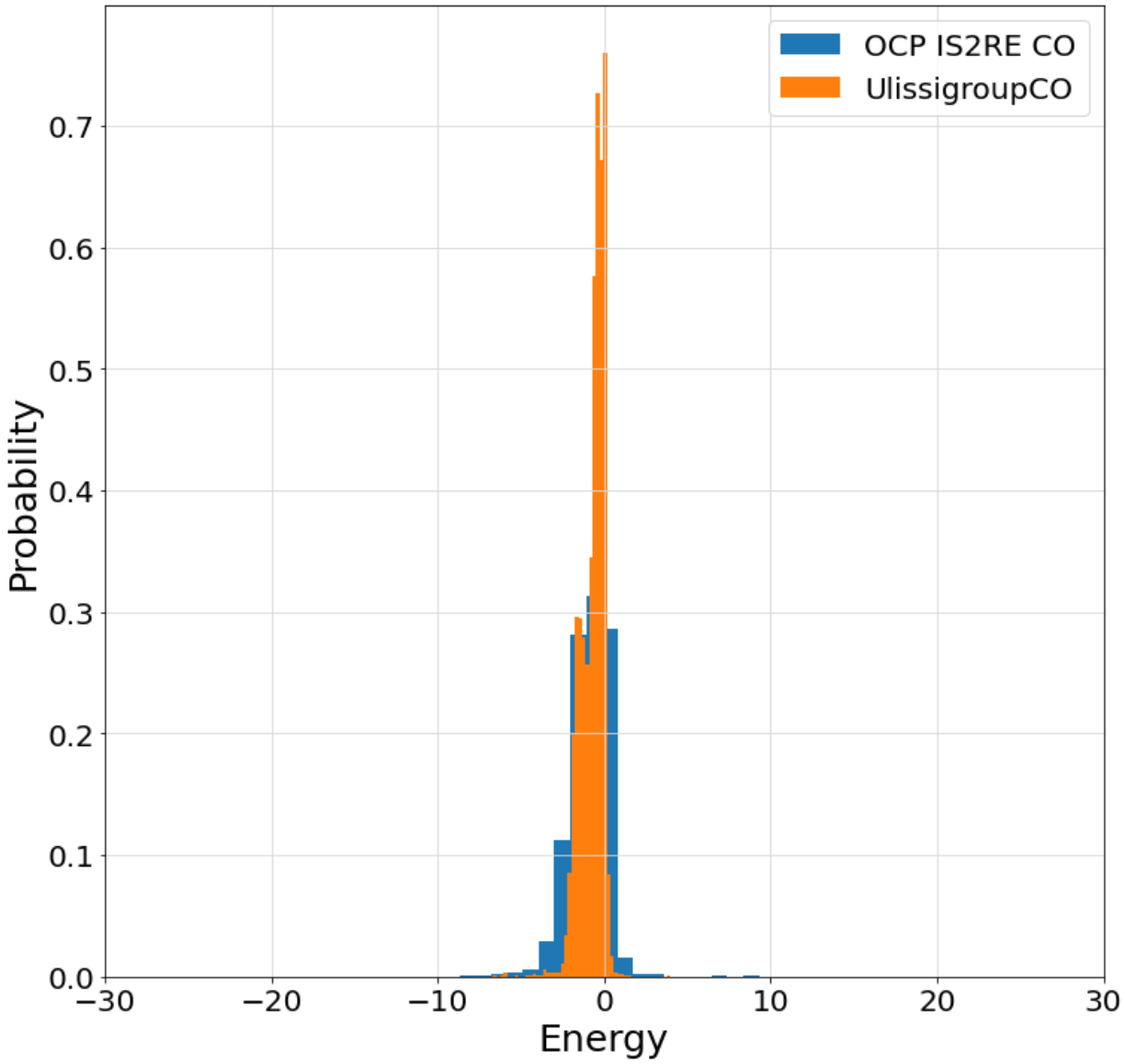
Out[291]:
```
(array([0.00661216, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.00661216, 0.        , 0.        ,
        0.        , 0.        , 0.00661216, 0.        , 0.00661214,
        0.03306072, 0.03967286, 0.09257017, 0.13224288, 0.18514004,
        0.30415863, 0.23803739, 0.35044364, 0.29754673, 0.24464933,
        0.17191582, 0.30415876, 0.54219582, 0.54219604, 0.68766327,
        0.75378466, 0.57525678, 1.17034987, 0.21820085, 0.06612146,
        0.        , 0.00661215, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.00661214, 0.        , 0.00661214]),
 array([-5.361466  , -5.21852   , -5.075574  , -4.932628  , -4.789682  ,
        -4.646736  , -4.5037904 , -4.360844  , -4.2178984 , -4.074952  ,
        -3.9320064 , -3.7890604 , -3.6461143 , -3.5031686 , -3.3602226 ,
        -3.2172766 , -3.0743306 , -2.9313846 , -2.7884388 , -2.6454928 ,
        -2.5025468 , -2.3596008 , -2.2166548 , -2.073709  , -1.930763  ,
        -1.787817  , -1.644871  , -1.5019251 , -1.3589791 , -1.2160332 ,
        -1.0730872 , -0.93014127, -0.7871953 , -0.6442493 , -0.5013034 ,
        -0.35835743, -0.21541147, -0.07246552,  0.07048044,  0.2134264 ,
         0.35637236,  0.4993183 ,  0.64226425,  0.78521025,  0.9281562 ,
         1.0711021 ,  1.2140481 ,  1.356994  ,  1.49994   ,  1.6428859 ,
         1.7858319 ], dtype=float32),
 <a list of 50 Patch objects>)
```


1.2
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4

In [225]:
```
plt.scatter(list(range(len(energies))), energies, label="OCP IS2RE CO")
plt.scatter(list(range(len(uco_energies))), uco_energies, label="Ulissigrou

plt.grid(color="0.85")
plt.ylim(-30, 30)
plt.xlim(0, )

plt.ylabel("Energy")
plt.xlabel("Data id")

plt.legend()

# plt.savefig(os.path.join("figures", "co_energy_dist_1"),
#             dpi=150,
#             bbox_inches="tight")
```

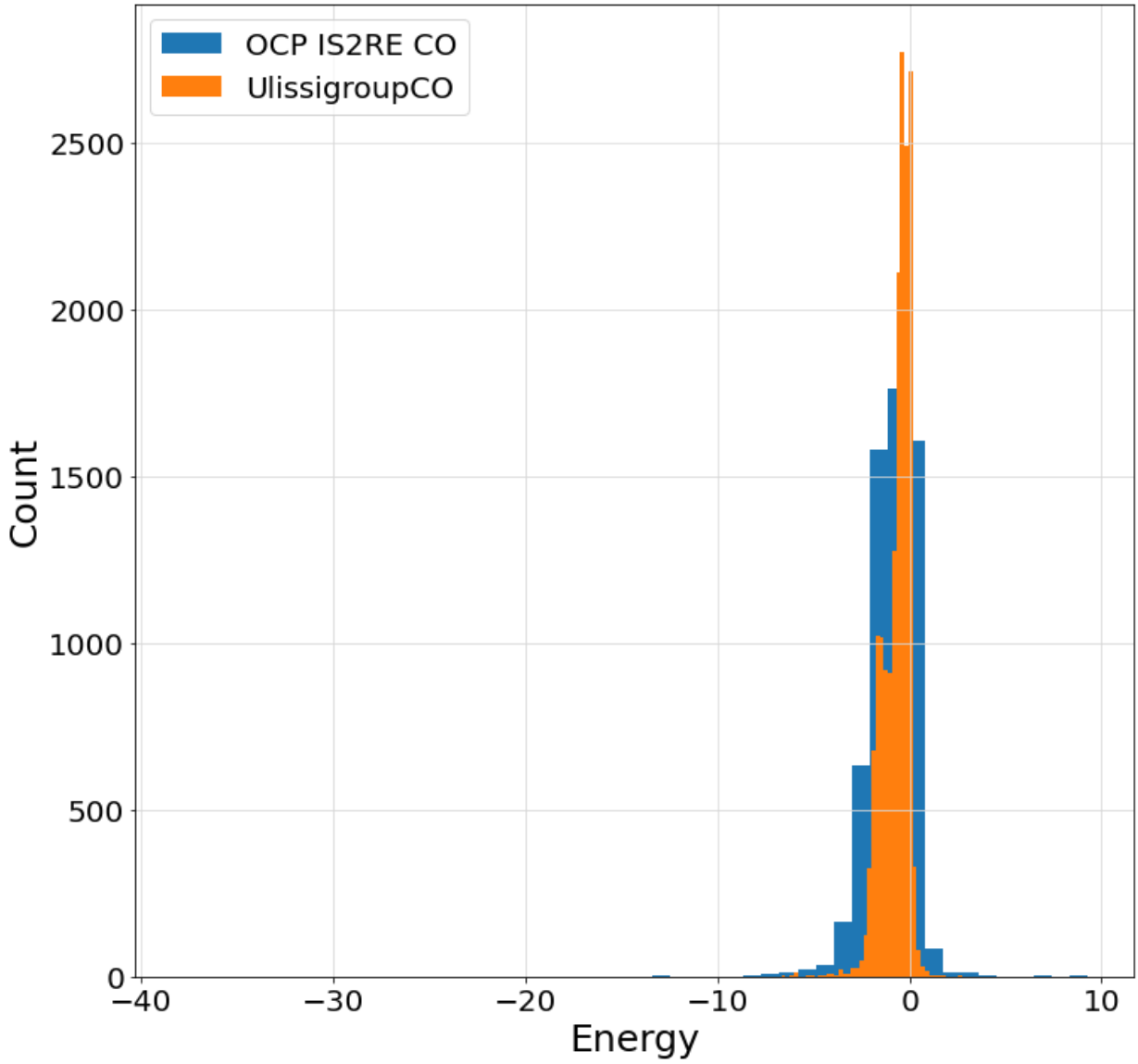


In [227]:
```
plt.hist(energies.flatten(), bins=50, density=True, label="OCP IS2RE CO")
# plt.hist(uco_init_data[0].y, bins=50, density=True, label="UlissigroupCO"
plt.hist(uco_energies.flatten(), bins=50, density=True, label="UlissigroupC

plt.grid(color="0.85")
plt.xlim(-30, 30)

plt.ylabel("Probability")
plt.xlabel("Energy")

plt.legend()

plt.savefig(os.path.join("figures", "co_energy_dist_2"),
            dpi=150,
            bbox_inches="tight")
```



In [70]:
```
plt.hist(energies.flatten(), bins=50, label="OCP IS2RE CO")
plt.hist(uco_init_data[0].y, bins=50, label="UlissigroupCO")
plt.grid(color="0.85")

plt.ylabel("Count")
plt.xlabel("Energy")

plt.legend()
```

Out[70]: <matplotlib.legend.Legend at 0x7f260905d0f0>



In [71]:
```
data, slices = uco_init_data
print(data)
```

```
Data(edge_attr=[7661772, 6], edge_index=[2, 7661772], x=[638481, 98], y=
[17019])
```

In [73]:
```
from ocpmodels.datasets import UlissigroupCO

uco_config = {
    "src": os.path.join(root_dir, "data/data/2020_02_16_ulissigroup_co/init
    "train_size": 14000,
    "val_size": 1000,
    "test_size": 1000,
}

uco_dataset = UlissigroupCO(uco_config)
uco_train, uco_val, uco_test = uco_dataset.get_dataloaders(batch_size=256)
```

In [75]:
```
print(uco_dataset[0])
print(dataset[0])
```

```
Data(edge_attr=[408, 6], edge_index=[2, 408], x=[34, 98], y=[1])
Data(atomic_numbers=[34], cell=[1, 3, 3], cell_offsets=[408, 3], distance
s=[408], edge_index=[2, 408], fixed=[34], force=[34, 3], natoms=34, pos=
[34, 3], tags=[34], y_init=3.141588459999994, y_relaxed=-0.12879014000000
666)
```

In [79]:
```
import pickle

pkl = pickle.load(open(os.path.join(root_dir, "data/data/2020_02_16_ulissig
```

In [183]:
```
uco_dataset[0].edge_index
```

Out[183]:
```
tensor([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,
          1,  1,
          1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  2,
          2,  2,
          3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4,
          4,  4,
          4,  4,  4,  4,  4,  4,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,
          5,  5,
          6,  6,  6,  6,  6,  6,  6,  6,  6,  6,  6,  6,  7,  7,  7,  7,
          7,  7,
          7,  7,  7,  7,  7,  7,  8,  8,  8,  8,  8,  8,  8,  8,  8,  8,
          8,  8,
          9,  9,  9,  9,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10,
         10, 10,
         10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
         11, 11,
         12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
         13, 13,
         13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
         14, 14,
         15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16,
         16, 16,
         16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
         17, 17,
         18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
         19, 19,
         19, 19, 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
         20, 20,
         21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
         22, 22,
         22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
         23, 23,
         24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 25, 25, 25, 25,
         25, 25,
         25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
         26, 26,
         27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28,
         28, 28,
         28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
         29, 29,
         30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 31, 31, 31, 31,
         31, 31,
         31, 31, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
         32, 32,
         33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33],
        [ 6, 33, 19,  5, 26, 11, 23,  1, 11,  0,  0,  0,  1,  1, 23,  1,
          1,  1,
          1, 11, 15,  0, 15, 33, 18, 32, 30, 22, 29, 24,  9,  7,  4,  3,
         16, 14,
         28, 19, 23, 31, 30, 22,  8,  9,  5,  2, 14, 16, 27, 32, 25, 20,
         24,  7,
         15, 17,  2, 16, 21, 15, 23, 33, 26, 31,  0,  8,  6,  3, 19, 14,
         10, 19,
          0, 19, 33, 26, 18, 22,  5,  8, 11, 10, 11, 24, 27, 21, 32, 29,
          4, 13,
```

```
          2, 15, 13,  9, 12, 15, 26, 24, 31, 20, 28, 18,  3,  5,  9,  6,
12, 10,
         28, 29, 25, 30, 21, 20,  2,  3,  8,  7, 13, 12, 22, 20, 24, 31,
33, 28,
         17, 16, 12, 11,  6,  8, 23, 26, 33, 22, 18, 14, 16, 10,  0,  6,
1, 19,
         25, 24, 21, 31, 30, 29, 17, 15, 13, 10,  8,  9, 15, 32, 27, 25,
30,  7,
         12,  7,  9, 21, 15, 15, 19, 23, 28, 18, 31, 29, 11, 16,  3,  5,
33,  2,
         21, 13, 25, 17, 12,  4, 32,  7, 15, 15,  1, 13, 18, 22, 20, 29,
27, 30,
         10, 11, 17,  4,  2, 14, 21, 32, 27, 20, 24, 15, 12, 10,  4, 16,
4,  2,
         16, 14,  2, 11,  6,  8, 22, 24, 26, 19, 20, 29, 14, 23,  6,  3,
0, 22,
         18, 28, 23,  5, 26, 33, 16, 17, 10,  4,  8,  9, 25, 24, 28, 21,
18, 27,
         15,  7, 17, 25, 12,  9, 29, 20, 13,  4, 15, 13, 10, 16, 11,  2,
6,  3,
         19, 18, 30, 24, 23, 33,  5, 11, 14, 19,  3,  1, 31, 19,  0, 22,
33,  0,
         12,  8, 10, 17,  4,  2, 20, 18, 32, 22, 25, 31, 15,  4, 13, 21,
12,  9,
         20, 24, 30, 17, 15, 17, 33,  5, 11,  8,  6, 28,  0, 18, 19, 10,
3,  0,
          7,  4, 13, 17, 16, 32, 30, 29, 15, 20, 15,  0, 14,  3,  9,  8,
10, 33,
         31, 26, 20, 29, 19,  2,  7,  9,  2, 16, 12, 31, 14, 30, 21, 28,
27, 18,
          2,  9,  3, 13, 12, 16, 27, 29, 22, 31, 32, 25, 12,  8, 10,  3,
5, 14,
         29, 28, 23, 30, 33, 24, 13,  4,  7, 17,  2, 27, 15, 24, 30, 16,
0,  0,
          5, 26, 11,  6, 10,  0, 28, 31, 19, 14, 22,  1]])
```

In [204]:
```
uco_dir = "/private/home/abhshkdz/projects/ocp-baselines/data/data/2020_08_

energies = pickle.load(open(os.path.join(uco_dir, "energy_keys.pkl"), "rb")
print(len(energies.keys()))
```

```
19087
```

In [193]:
```
trajs = glob.glob(uco_dir + "/trajs/*traj")
trajs = [int(i.split("/")[-1].split(".")[0]) for i in trajs]
print(trajs)

energy_keys = list(energies.keys())
print(energy_keys)
```

```
False

/private/home/abhshkdz/.conda/envs/ocp-models/lib/python3.6/site-package
s/ipykernel_launcher.py:8: DeprecationWarning: elementwise comparison fai
led; this will raise an error in the future.
```

Debugging PBC

In [85]:
```
from ocpmodels.preprocessing import AtomsToGraphs
```

In [167]:
```
traj = Trajectory("/checkpoint/sidgoyal/electro_done/random1623153.traj")

a2g = AtomsToGraphs(
    max_neigh=12,
    radius=6,
    dummy_distance=7,
    dummy_index=-1,
    r_energy=True,
    r_forces=True,
    r_distances=True,
)

data = a2g.convert(traj[768])
```

In [168]:
```
large = torch.tensor([984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994

print(data.distances[large])
print(data.edge_index[:, large])
```

```
tensor([5.1633, 3.3166, 1.0877, 2.1362, 3.1236, 4.0045, 5.9650, 4.6918,
7.0000,
        7.0000, 7.0000, 7.0000])
tensor([[82, 82, 82, 82, 82, 82, 82, 82, 82, 82, 82, 82],
        [32, 84, 81, 80, 29, 42, 77, 56, -1, -1, -1, -1]])
```

In [169]:
```
data.cell_offsets[large]
```

Out[169]:
```
tensor([[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 1, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]])
```

In [170]:
```
row, col = data.edge_index

distance_vectors = data.pos[row] - data.pos[col]

cell = torch.repeat_interleave(data.cell, data.natoms * 12, dim=0)
offsets = data.cell_offsets.float().view(-1, 1, 3).bmm(cell.float()).view(-
distance_vectors -= offsets
```

In [171]:
```
distances = distance_vectors.norm(dim=-1)

print(data.distances[:10])
print(distances[:10])

print(data.distances[large])
print(distances[large])
```

```
tensor([3.9221, 3.0545, 3.8578, 4.2377, 5.5366, 4.2819, 4.1080, 7.0000,
7.0000,
        7.0000])
tensor([ 3.9221,  3.0545,  3.8578,  4.2377,  5.5366,  4.2819,  4.1080, 1
3.5885,
        13.5885, 13.5885])
tensor([5.1633, 3.3166, 1.0877, 2.1362, 3.1236, 4.0045, 5.9650, 4.6918,
7.0000,
        7.0000, 7.0000, 7.0000])
tensor([29.5358, 30.5126, 31.7990, 31.1695, 30.3438, 31.2990, 30.7791, 3
2.9409,
        30.5126, 30.5126, 30.5126, 30.5126])
```

In [114]:
```
(data.pos[82] - data.pos[32]).norm()
```

Out[114]:
```
tensor(29.5358)
```

In [117]:
```
from pymatgen.io.ase import AseAtomsAdaptor

struct = AseAtomsAdaptor.get_structure(traj[768])
print(struct)
```

```
Full Formula (Rb48 Bi32 H2 C2 O1)
Reduced Formula: Rb48Bi32H2C2O
abc   :  14.642170  20.091146  43.926510
angles:  98.917017  88.078529  81.082983
Sites (85)
  #  SP           a         b         c
---  ----  --------  --------  --------
  0  Bi    0.431733  0.578841  0.35344
  1  Bi    0.431733  0.578841  0.520107
  2  Bi    0.175793  0.084227  0.600954
  3  Bi    0.181733  0.078841  0.436773
  4  Bi    0.378135  0.921159  0.513074
  5  Bi    0.378135  0.921159  0.346408
  6  Bi    0.128135  0.421159  0.429741
  7  Bi    0.128135  0.421159  0.596408
  8  Bi    0.290482  0.671159  0.400523
  9  Bi    0.290482  0.671159  0.56719
 10  Bi    0.063544  0.179313  0.652535
 11  Bi    0.040482  0.171159  0.483857
 12  Bi    0.019385  0.828841  0.465991
 13  Bi    0.010024  0.819271  0.632501
 14  Bi    0.269385  0.328841  0.382658
 15  Bi    0.269385  0.328841  0.549324
 16  Bi    0.931733  0.578841  0.35344
 17  Bi    0.931733  0.578841  0.520107
 18  Bi    0.684693  0.083517  0.600608
 19  Bi    0.681733  0.078841  0.436773
 20  Bi    0.878135  0.921159  0.513074
 21  Bi    0.878135  0.921159  0.346408
 22  Bi    0.628135  0.421159  0.429741
 23  Bi    0.628135  0.421159  0.596408
 24  Bi    0.790482  0.671159  0.400523
 25  Bi    0.790482  0.671159  0.56719
 26  Bi    0.56417   0.174206  0.651425
 27  Bi    0.540482  0.171159  0.483857
 28  Bi    0.519385  0.828841  0.465991
 29  Bi    0.583531  0.807834  0.632074
 30  Bi    0.769385  0.328841  0.382658
 31  Bi    0.769385  0.328841  0.549324
 32  Rb    0.239178  0.755034  0.658345
 33  Rb    0.288413  0.737688  0.487819
 34  Rb    0.038413  0.237688  0.571152
 35  Rb    0.038413  0.237688  0.404486
 36  Rb    0.021455  0.762312  0.545362
 37  Rb    0.021455  0.762312  0.378696
 38  Rb    0.271455  0.262312  0.462029
 39  Rb    0.296283  0.235789  0.627024
 40  Rb    0.193619  0.512312  0.35275
 41  Rb    0.193619  0.512312  0.519417
 42  Rb    0.475359  0.000922  0.628769
 43  Rb    0.443619  0.012312  0.436084
 44  Rb    0.116249  0.987688  0.513764
```

45	Rb	0.116249	0.987688	0.347098
46	Rb	0.366249	0.487688	0.430431
47	Rb	0.360151	0.486477	0.593118
48	Rb	0.05113	0.625	0.440323
49	Rb	0.049642	0.626716	0.603304
50	Rb	0.30113	0.125	0.356989
51	Rb	0.30113	0.125	0.523656
52	Rb	0.258737	0.875	0.426192
53	Rb	0.230078	0.881962	0.592742
54	Rb	0.050554	0.370482	0.671327
55	Rb	0.008737	0.375	0.509525
56	Rb	0.807309	0.778186	0.672928
57	Rb	0.788413	0.737688	0.487819
58	Rb	0.538413	0.237688	0.571152
59	Rb	0.538413	0.237688	0.404486
60	Rb	0.521455	0.762312	0.545362
61	Rb	0.521455	0.762312	0.378696
62	Rb	0.771455	0.262312	0.462029
63	Rb	0.795251	0.240236	0.626934
64	Rb	0.693619	0.512312	0.35275
65	Rb	0.693619	0.512312	0.519417
66	Rb	0.959065	0.015577	0.638838
67	Rb	0.943619	0.012312	0.436084
68	Rb	0.616249	0.987688	0.513764
69	Rb	0.616249	0.987688	0.347098
70	Rb	0.866249	0.487688	0.430431
71	Rb	0.860112	0.486563	0.593451
72	Rb	0.55113	0.625	0.440323
73	Rb	0.553131	0.623948	0.605397
74	Rb	0.80113	0.125	0.356989
75	Rb	0.80113	0.125	0.523656
76	Rb	0.758737	0.875	0.426192
77	Rb	0.779393	0.881737	0.593424
78	Rb	0.545631	0.366948	0.670513
79	Rb	0.508737	0.375	0.509525
80	C	0.393292	0.878513	0.664678
81	C	0.477168	0.871884	0.673331
82	H	0.031913	0.112866	0.017937
83	H	0.500246	0.887369	0.696253
84	O	0.317313	0.882655	0.653916

In [127]:
```
split_n_index, split_n_distances, split_offsets = a2g._get_neighbors_pymatg
```

In [138]:
```
print(split_n_index[82])
print(split_n_distances[82])
print(split_offsets[82])
```

```
[32 84 81 80 29 42 77 56]
[5.16325855 3.3166412  1.08769144 2.13621303 3.1236147  4.00451069
 5.96501703 4.69182041]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 1. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [149]:
```
print(struct[32])
print(struct[82])
```

```
[ 5.93164618 14.78757356 26.68489623] Rb
[0.83049532 2.21052316 0.45209488] H
```

In [161]:
```
print(struct.sites[32].coords)
print(struct.sites[82].coords)
```

```
[ 5.93164618 14.78757356 26.68489623]
[0.83049532 2.21052316 0.45209488]
```

In [182]:
```
(data.distances == distances).sum()
```

Out[182]:
```
tensor(275)
```

max_nbr vs. no. of edges and triplets

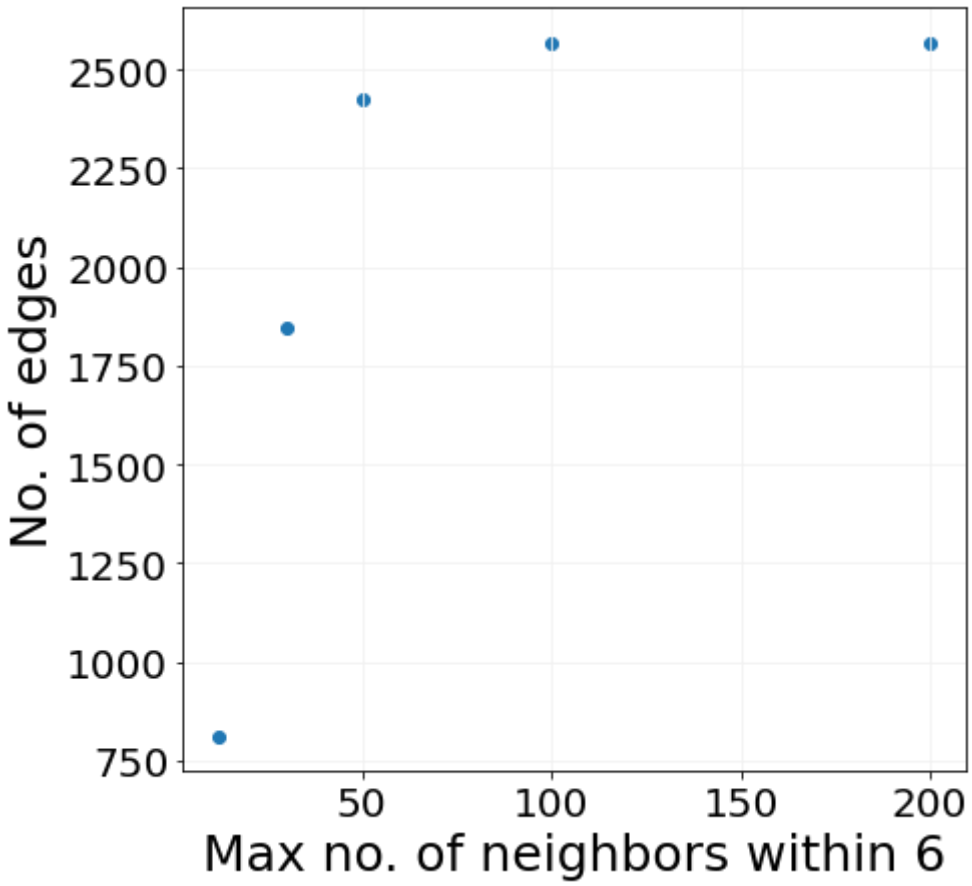
In [60]:
```
from torch.utils.data import DataLoader
from ocpmodels.common.utils import get_pbc_distances
from ocpmodels.datasets import SinglePointLmdbDataset, data_list_collater

from torch_sparse import SparseTensor

src = "../data/data/2020_08_18_ocp_co_is2re/max_nbr_12/train/data.lmdb"

dataset_config = {
    "src": src,
}
dataset = SinglePointLmdbDataset(dataset_config)
data_loader = DataLoader(
    dataset,
    batch_size=256,
    shuffle=False,
    collate_fn=data_list_collater,
    num_workers=16,
)


num_atoms, num_edges, num_angles = [], [], []


for i, batch in tqdm(enumerate(data_loader)):
    num_atoms += batch.natoms.tolist()
    num_edges += batch.neighbors.tolist()

    edge_index, dist, offsets = get_pbc_distances(
        batch.pos,
        batch.edge_index,
        batch.cell,
        batch.cell_offsets,
        batch.neighbors,
        6.0,
        return_offsets=True,
    )

    row, col = edge_index  # j->i
    num_nodes = batch.atomic_numbers.size(0)

    value = torch.arange(row.size(0), device=row.device)
    adj_t = SparseTensor(row=col, col=row, value=value,
                         sparse_sizes=(num_nodes, num_nodes))
    adj_t_row = adj_t[row]
    num_triplets = adj_t_row.set_value(None).sum(dim=1).to(torch.long)
    num_angles.append(num_triplets.sum().item())
```

```
31it [00:06,  4.97it/s]
```

In [69]:
```
print("num_atoms", np.mean(num_atoms))
print("num_edges", np.mean(num_edges))
print("num_angles", np.sum(num_angles) / len(num_edges))
```

```
num_atoms 68.8485824742268
num_edges 812.7167525773195
num_angles 9553.124613402062
```

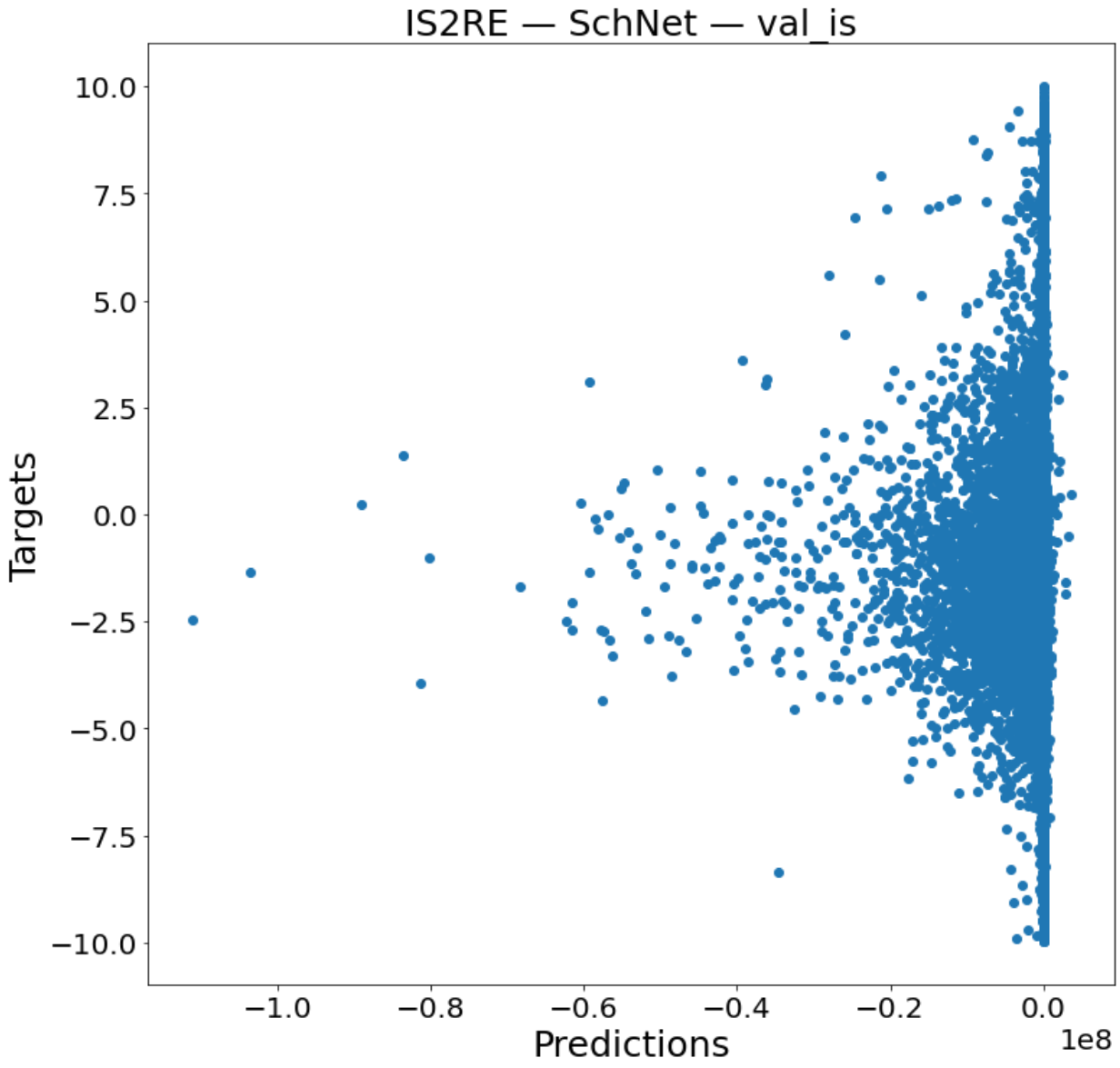
In [96]:
```
x = [12, 30, 50, 100, 200]
num_edges = [812.71, 1846.84, 2423.93, 2567.29, 2569.45]
num_angles = [9553.12, 51246.02, 94911.75, 111818.33, 112272.38]


fig = plt.figure(figsize=(7, 7))

plt.scatter(x, num_edges)
# plt.scatter(x, num_angles)
plt.grid(color="0.95")

plt.xlabel("Max no. of neighbors within 6")
plt.ylabel("No. of edges")

# plt.savefig(os.path.join("figures", "nbr_edges.png"),
#             dpi=150,
#             bbox_inches="tight")
```



IS2RE analysis

In [13]:
```
# val_is
predictions = np.load("/private/home/abhshkdz/projects/ocp-baselines/predic
targets = np.load("/private/home/abhshkdz/projects/ocp-baselines/prediction
```

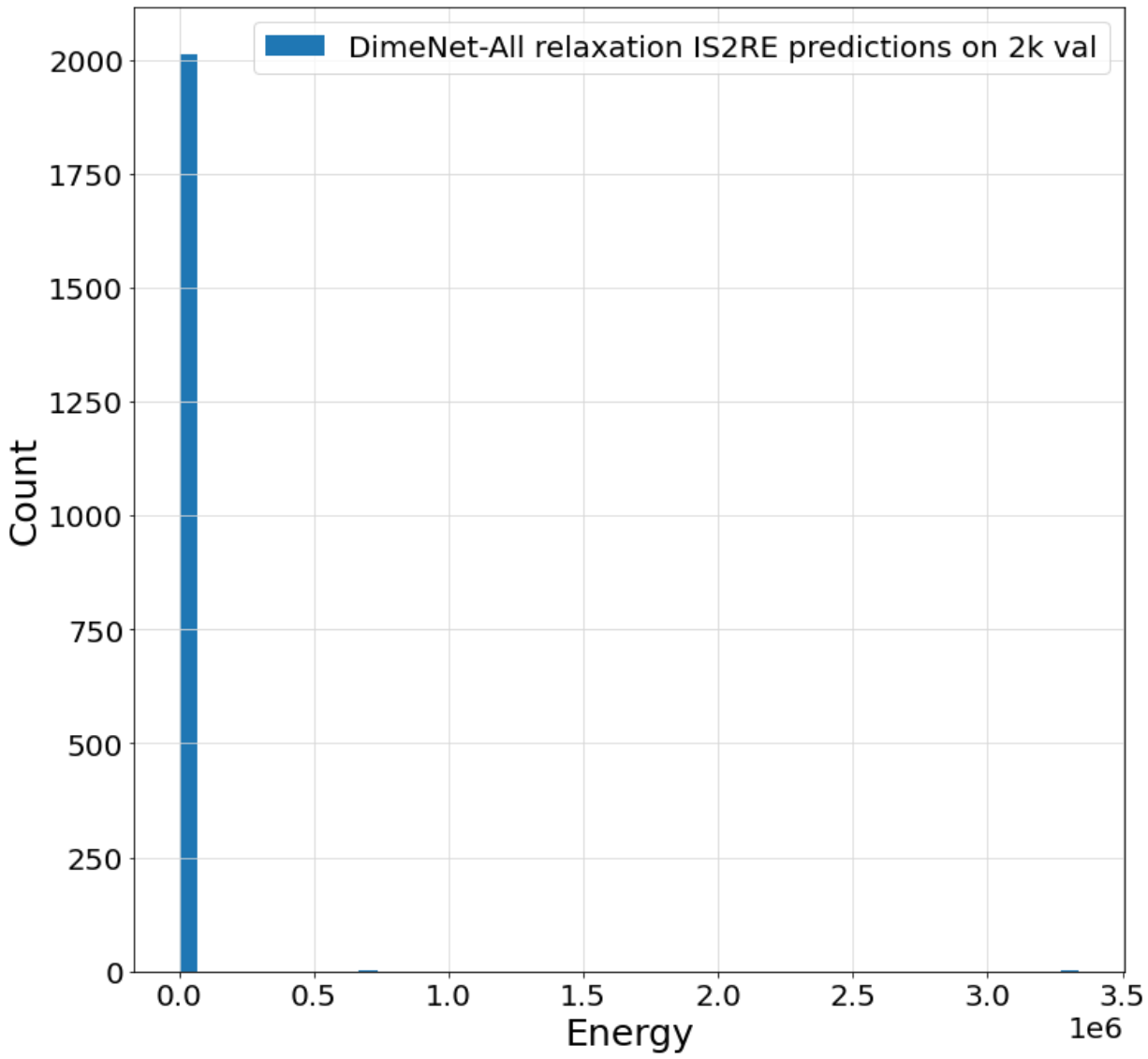
In [14]:
```
plt.scatter(predictions, targets)
plt.xlabel("Predictions")
plt.ylabel("Targets")
plt.title("IS2RE — SchNet — val_is")

plt.savefig(os.path.join("figures", "cgcnn_is2re_preds.png"),
            dpi=150,
            bbox_inches="tight")
```



In [3]:
```
e_preds = json.load(open("/private/home/abhshkdz/projects/ocp-baselines/ext
e_preds = np.array(e_preds)
```

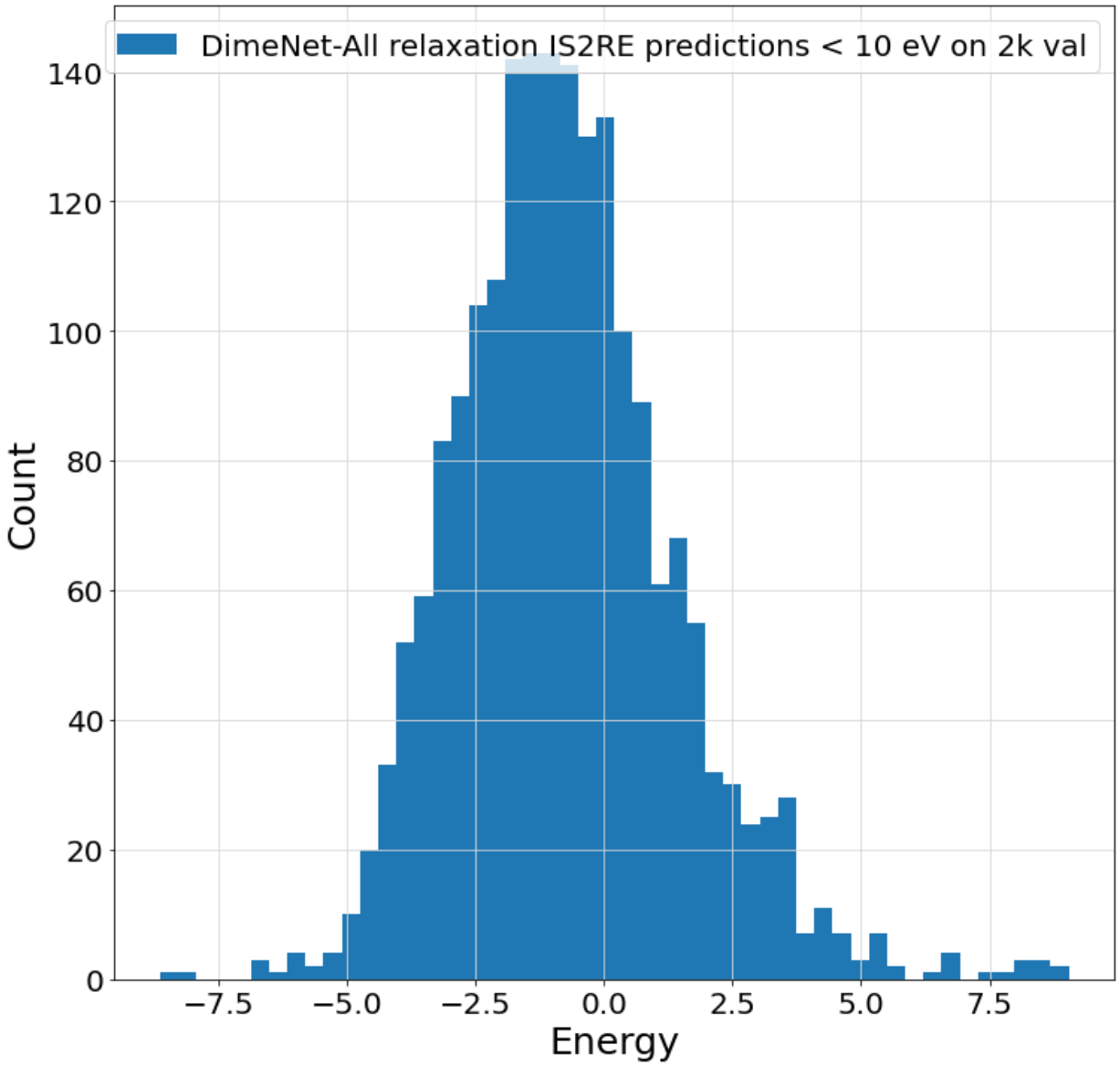
In [12]:
```
plt.hist(e_preds, bins=50, label="DimeNet-All relaxation IS2RE predictions
plt.grid(color="0.85")

plt.ylabel("Count")
plt.xlabel("Energy")

plt.legend()
```

Out[12]: <matplotlib.legend.Legend at 0x7fef220329b0>



In [13]:
```
plt.hist(e_preds[np.where(e_preds < 10)], bins=50, label="DimeNet-All relax
plt.grid(color="0.85")

plt.ylabel("Count")
plt.xlabel("Energy")

plt.legend()
```

Out[13]: <matplotlib.legend.Legend at 0x7fef9cd49358>



In [14]:
```
e_preds[np.where(e_preds > 10)]
```

Out[14]:
```
array([1.22831602e+01, 2.32600296e+02, 5.76015053e+01, 4.88243675e+01,
       4.80988731e+01, 2.04663538e+03, 4.16718231e+02, 6.89577562e+05,
       1.29277363e+04, 1.69536118e+02, 2.05103207e+01, 1.68791409e+01,
       1.17875328e+01, 8.28141663e+02, 1.45897446e+01, 9.84662720e+02,
       9.79078522e+01, 2.14943774e+03, 4.37249374e+01, 1.76819382e+01,
       1.21827679e+01, 1.03473301e+01, 2.29007721e+01, 9.12189392e+02,
       7.76929871e+02, 3.34000425e+06, 2.47430439e+01, 3.79839336e+04,
       3.42124100e+01, 2.17648828e+04, 8.92485938e+03, 1.22666645e+01,
       4.60435078e+04, 1.51534814e+03, 1.50058258e+02, 1.32583303e+01,
       3.82320900e+01, 4.58789444e+01, 1.56265579e+02, 3.55418015e+01,
       1.53786907e+01, 6.40656396e+03, 5.67351257e+02, 1.64219580e+03])
```

Prediction file sizes

In [19]:
```
pred_file = "/checkpoint/abhshkdz/s2ef_eval/npz/s2ef_schnet_all_evalai_subm

preds = np.load(pred_file, allow_pickle=True)
```

In [20]:
```
list(preds.keys())
```

Out[20]:
```
['id_ids',
 'id_energy',
 'id_forces',
 'ood_ads_ids',
 'ood_ads_energy',
 'ood_ads_forces',
 'ood_cat_ids',
 'ood_cat_energy',
 'ood_cat_forces',
 'ood_both_ids',
 'ood_both_energy',
 'ood_both_forces']
```

In [22]:
```
preds["id_forces"][0].dtype
```

Out[22]:
```
dtype('float32')
```

In [30]:
```
preds_dict = dict(preds)

for i in preds_dict:
    if "energy" in i:
        preds_dict[i] = np.float16(preds_dict[i])
    elif "forces" in i:
        for j in range(len(preds[i])):
            preds_dict[i][j] = np.float16(preds_dict[i][j])
```

In [31]:
```
np.savez_compressed("/checkpoint/abhshkdz/s2ef_eval/npz/s2ef_schnet_all_eva
```

In [33]:
```
np.save("/checkpoint/abhshkdz/s2ef_eval/npz/s2ef_schnet_all_evalai_submissi
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-33-cb8c538b4950> in <module>
----> 1 np.save("/checkpoint/abhshkdz/s2ef_eval/npz/s2ef_schnet_all_evala
i_submission_raw_float16.npy", **preds_dict)

<__array_function__ internals> in save(*args, **kwargs)

TypeError: _save_dispatcher() got an unexpected keyword argument 'id_ids'
```

IS2RS run using MD + 20M

- id: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-01-28-44-evaltest_is/relaxed_positions.npz
- ood_ads: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-05-02-evaltest_oos_ads/relaxed_positions.npz
- ood_cat: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-01-28-44-evaltest_oos_bulk/relaxed_positions.npz
- ood_both: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-05-02-evaltest_oos_ads_bulk/relaxed_positions.npz

IS2RS run using Rattled + 20M

- id: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-08-31-evaltest_is/relaxed_positions.npz
- ood_ads: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-21-12-evaltest_oos_ads/relaxed_positions.npz
- ood_cat: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-26-48-evaltest_oos_bulk/relaxed_positions.npz
- ood_both: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-26-20-29-23-evaltest_oos_ads_bulk/relaxed_positions.npz

IS2RS run using SchNet-large on S2EF all

- id: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-44-evaltest_is/relaxed_positions.npz
- ood_ads: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-26-53-evaltest_oos_ads/relaxed_positions.npz
- ood_cat: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-36-32-evaltest_oos_bulk/relaxed_positions.npz
- ood_both: /private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-42-57-evaltest_oos_ads_bulk/relaxed_positions.npz

In [123]:
```
import glob
from collections import defaultdict

npz_files = glob.glob("/private/home/abhshkdz/projects/ocp-baselines/result
print(len(npz_files), npz_files)
```

```
32 ['/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-
19-44-evaltest_is/relaxed_pos_0.npz', '/private/home/abhshkdz/projects/oc
p-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_1.npz',
'/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-4
3-evaltest_is/relaxed_pos_5.npz', '/private/home/abhshkdz/projects/ocp-ba
selines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_4.npz', '/pri
vate/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-43-eva
ltest_is/relaxed_pos_6.npz', '/private/home/abhshkdz/projects/ocp-baselin
es/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_7.npz', '/private/
home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-43-evaltest
_is/relaxed_pos_8.npz', '/private/home/abhshkdz/projects/ocp-baselines/re
sults/2020-10-28-18-19-43-evaltest_is/relaxed_pos_3.npz', '/private/home/
abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/r
elaxed_pos_10.npz', '/private/home/abhshkdz/projects/ocp-baselines/result
s/2020-10-28-18-19-43-evaltest_is/relaxed_pos_9.npz', '/private/home/abhs
hkdz/projects/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/relax
ed_pos_13.npz', '/private/home/abhshkdz/projects/ocp-baselines/results/20
20-10-28-18-19-43-evaltest_is/relaxed_pos_15.npz', '/private/home/abhshkd
z/projects/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_
pos_14.npz', '/private/home/abhshkdz/projects/ocp-baselines/results/2020-
10-28-18-19-43-evaltest_is/relaxed_pos_12.npz', '/private/home/abhshkdz/p
rojects/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos
_24.npz', '/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-
28-18-19-43-evaltest_is/relaxed_pos_11.npz', '/private/home/abhshkdz/proj
ects/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_2
6.npz', '/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28
-18-19-43-evaltest_is/relaxed_pos_2.npz', '/private/home/abhshkdz/project
s/ocp-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_25.np
z', '/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-
19-43-evaltest_is/relaxed_pos_28.npz', '/private/home/abhshkdz/projects/o
cp-baselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_31.npz',
'/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-4
3-evaltest_is/relaxed_pos_29.npz', '/private/home/abhshkdz/projects/ocp-b
aselines/results/2020-10-28-18-19-43-evaltest_is/relaxed_pos_30.npz', '/p
rivate/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-43-e
valtest_is/relaxed_pos_27.npz', '/private/home/abhshkdz/projects/ocp-base
lines/results/2020-10-28-18-19-50-evaltest_is/relaxed_pos_16.npz', '/priv
ate/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-50-eval
test_is/relaxed_pos_18.npz', '/private/home/abhshkdz/projects/ocp-baselin
es/results/2020-10-28-18-19-50-evaltest_is/relaxed_pos_20.npz', '/privat
e/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-50-evalte
st_is/relaxed_pos_23.npz', '/private/home/abhshkdz/projects/ocp-baseline
s/results/2020-10-28-18-19-50-evaltest_is/relaxed_pos_22.npz', '/private/
home/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-50-evaltest
_is/relaxed_pos_17.npz', '/private/home/abhshkdz/projects/ocp-baselines/r
esults/2020-10-28-18-19-50-evaltest_is/relaxed_pos_21.npz', '/private/hom
e/abhshkdz/projects/ocp-baselines/results/2020-10-28-18-19-50-evaltest_i
s/relaxed_pos_19.npz']
```

In [124]:
```
gather_results = defaultdict(list)

for i in npz_files:
    f = np.load(i, allow_pickle=True)
    gather_results["ids"].extend(f["ids"])
    gather_results["pos"].extend(f["pos"])

print(len(gather_results["ids"]))
print(len(gather_results["pos"]))

np.savez("/private/home/abhshkdz/projects/ocp-baselines/results/2020-10-28-
```

```
24960
24960
```

In [72]:
```
a = np.load("/checkpoint/abhshkdz/is2rs_eval/predictions/test/10_26/schnet_
print(list(a.keys()))

print(len(a["id_ids"]))
print(len(a["ood_ads_ids"]))
print(len(a["ood_cat_ids"]))
print(len(a["ood_both_ids"]))

print(len(list(set(a["id_ids"])))))
print(len(list(set(a["ood_ads_ids"])))))
print(len(list(set(a["ood_cat_ids"])))))
print(len(list(set(a["ood_both_ids"])))))

# print(a["id_pos"][0])

# print(a["id_ids"][:5])
# print(a["ood_ads_ids"][:5])

print(len(list(set(a["id_ids"].tolist() + a["ood_ads_ids"].tolist() + a["oo
print(len(a["id_ids"].tolist() + a["ood_ads_ids"].tolist() + a["ood_cat_ids
```

```
['id_ids', 'id_pos', 'ood_ads_ids', 'ood_ads_pos', 'ood_cat_ids', 'ood_ca
t_pos', 'ood_both_ids', 'ood_both_pos']
24960
24960
24992
24992
24951
24931
24967
24986
99835
99904
```

In [76]:
```
a = np.load("/checkpoint/abhshkdz/is2re_eval/predictions/val/10_27/dimenet_
print(list(a.keys()))

print(a["id_energy"].shape)
print(a["ood_ads_energy"].shape)
print(a["ood_cat_energy"].shape)
print(a["ood_both_energy"].shape)
```

```
['id_ids', 'id_energy', 'ood_ads_ids', 'ood_ads_energy', 'ood_cat_ids',
'ood_cat_energy', 'ood_both_ids', 'ood_both_energy']
(24946,)
(24966,)
(24964,)
(24988,)
```

Removing duplicates from IS2RS predictions

In [131]:
```
from tqdm import tqdm

a = np.load("/checkpoint/abhshkdz/is2re_eval/predictions/val/10_27/dimenet_

print(list(a.keys()))

final_preds = {}

for split in ["id", "ood_ads", "ood_cat", "ood_both"]:
    print(split)
    _, idx = np.unique(a[split+"_ids"], return_index=True)
    final_preds[split+"_ids"] = a[split+"_ids"][idx]
    final_preds[split+"_energy"] = a[split+"_energy"][idx]
```

```
['id_ids', 'id_energy', 'ood_ads_ids', 'ood_ads_energy', 'ood_cat_ids',
'ood_cat_energy', 'ood_both_ids', 'ood_both_energy']
id
ood_ads
ood_cat
ood_both
```

In [132]:
```
np.savez_compressed("/checkpoint/abhshkdz/is2re_eval/predictions/val/10_27/
```

In [1]:
```
a = np.load("/checkpoint/abhshkdz/is2re_eval/predictions/val/10_27/dimenet_
print(len(a["id_ids"]))
print(len(list(set(a["id_ids"]))))
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-223d2b70473d> in <module>
----> 1 a = np.load("/checkpoint/abhshkdz/is2re_eval/predictions/val/10_2
7/dimenet_all_dedup.npz", allow_pickle=True)
      2 print(len(a["id_ids"]))
      3 print(len(list(set(a["id_ids"]))))

NameError: name 'np' is not defined
```

S2EF val_id predictions from DimeNet++-Large-forceonly

In [8]:
```
def npz_2_s2ef_pred(npz_input_file: str, mode: str):

    with open(npz_input_file, "rb") as f:
        data = np.load(f)
        forces = data[f"{mode}_forces"]
        energy = data[f"{mode}_energy"]
        chunk_idx = data[f"{mode}_chunk_idx"]
        forces = np.split(forces, chunk_idx)
        ids = data[f"{mode}_ids"]

    energy = torch.tensor(energy)
    out_atoms = []
    for force_array in forces:
        out_atoms.append(force_array.shape[0])

    return {
        "energy": torch.tensor(energy),
        "forces": forces,
        "natoms": torch.tensor(out_atoms),
    }, ids

def npz_2_s2ef_pred_single_mode(npz_input_file: str):

    with open(npz_input_file, "rb") as f:
        data = np.load(f)
        forces = data[f"forces"]
        energy = data[f"energy"]
        chunk_idx = data[f"chunk_idx"]
        forces = np.split(forces, chunk_idx)
        ids = data[f"ids"]

    energy = torch.tensor(energy)
    out_atoms = []
    for force_array in forces:
        out_atoms.append(force_array.shape[0])

    return {
        "energy": torch.tensor(energy),
        "forces": forces,
        "natoms": torch.tensor(out_atoms),
    }, ids

def npz_2_s2ef_anno(npz_input_file: str, mode: str):

    with open(npz_input_file, "rb") as f:
        data = np.load(f, allow_pickle=True)
        forces = data[f"{mode}_forces"]
        energy = data[f"{mode}_energy"]
        ids = data[f"{mode}_ids"]

    energy = torch.tensor(energy)
    out_atoms = []
    for force_array in forces:
        out_atoms.append(force_array.shape[0])

    return {
```

```
        "energy": torch.tensor(energy),
        "forces": forces,
        "natoms": torch.tensor(out_atoms),
    }, ids

def reorder(ref: np.ndarray, to_reorder: np.ndarray) -> np.ndarray:
    assert len(ref) == len(set(ref))
    assert len(to_reorder) == len(set(to_reorder))
    assert set(ref) == set(to_reorder)
    item_to_idx = {item: idx for idx, item in enumerate(to_reorder)}
    return np.array([item_to_idx[item] for item in ref])

def check_ids_and_reorder(annotations_ids, predictions_ids, predictions):
    if set(predictions_ids) != set(annotations_ids):
        missing_ids = set(annotations_ids) - set(predictions_ids)
        unexpected_ids = set(predictions_ids) - set(annotations_ids)

        details = (
            f"{len(missing_ids)} missing IDs: ({list(missing_ids)[:3]}, ...
            f"{len(unexpected_ids)} unexpected IDs: ({list(unexpected_ids)[
        )
        import pdb; pdb.set_trace()

    order = reorder(annotations_ids, predictions_ids)

    predictions["natoms"] = predictions["natoms"][order]
    predictions["energy"] = predictions["energy"][order]

    out_forces = []
    for x in order:
        fa = predictions["forces"][x]
        out_forces.append(torch.tensor(fa))

    predictions["forces"] = torch.cat(out_forces, dim=0)

    return predictions
```

In [4]:
```
from ocpmodels.modules.evaluator import Evaluator

evaluator = Evaluator(task="s2ef")

preds, preds_ids = npz_2_s2ef_pred_single_mode("/private/home/abhshkdz/proj
annos, annos_ids = npz_2_s2ef_anno("/checkpoint/abhshkdz/open-catalyst-proj

preds = check_ids_and_reorder(annos_ids, preds_ids, preds)

out_forces = []
for force_array in annos["forces"]:
    out_forces.append(torch.tensor(force_array))
annos["forces"] = torch.cat(out_forces, dim=0)

metrics = evaluator.eval(preds, annos, prev_metrics={})
for key in metrics:
    print(key, metrics[key]["metric"])
```

```
/private/home/abhshkdz/.conda/envs/ocp-models/lib/python3.6/site-package
s/ipykernel_launcher.py:38: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or sourceTensor.cl
one().detach().requires_grad_(True), rather than torch.tensor(sourceTenso
r).
/private/home/abhshkdz/.conda/envs/ocp-models/lib/python3.6/site-package
s/ipykernel_launcher.py:57: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or sourceTensor.cl
one().detach().requires_grad_(True), rather than torch.tensor(sourceTenso
r).

forcesx_mae 0.02589291092893002
forcesy_mae 0.029048655982500143
forcesz_mae 0.029349668603046095
forces_mae 0.028097078504825422
forces_cos 0.563438940696577
forces_magnitude 0.032702871673278365
energy_mae 29.349254023324193
energy_force_within_threshold 2.1002814377126536e-05
```

Now that we've made sure we've loaded + evaluated predictions accurately, let's compute metrics per sample.

In [9]:
```
preds, preds_ids = npz_2_s2ef_pred_single_mode("/private/home/abhshkdz/proj
annos, annos_ids = npz_2_s2ef_anno("/checkpoint/abhshkdz/open-catalyst-proj
```

```
/private/home/abhshkdz/.conda/envs/ocp-models/lib/python3.6/site-package
s/ipykernel_launcher.py:38: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or sourceTensor.cl
one().detach().requires_grad_(True), rather than torch.tensor(sourceTenso
r).
/private/home/abhshkdz/.conda/envs/ocp-models/lib/python3.6/site-package
s/ipykernel_launcher.py:57: UserWarning: To copy construct from a tensor,
it is recommended to use sourceTensor.clone().detach() or sourceTensor.cl
one().detach().requires_grad_(True), rather than torch.tensor(sourceTenso
r).
```

In [10]:
```
order = reorder(annos_ids, preds_ids)
```

In [11]:
```
print(preds_ids[order])
print(annos_ids)
```

```
['2163697_34' '2099289_25' '1899087_116' ... '797361_41' '1332756_88'
 '2044112_78']
['2163697_34' '2099289_25' '1899087_116' ... '797361_41' '1332756_88'
 '2044112_78']
```

In [12]:
```
# MAKE SURE TO RUN ONLY ONCE!

reordered_preds = {
    "energy": [torch.tensor([k]) for k in preds["energy"][order]],
    "forces": [torch.from_numpy(preds["forces"][k]) for k in order],
    "natoms": [torch.tensor([k]) for k in preds["natoms"][order]],
}
reordered_preds_ids = preds_ids[order]

tensorized_annos = {
    "energy": [torch.tensor([k]) for k in annos["energy"]],
    "forces": [torch.from_numpy(k) for k in annos["forces"]],
    "natoms": [torch.tensor([k]) for k in annos["natoms"]],
}

assert reordered_preds_ids[42] == annos_ids[42]
```

In [13]:
```
len(reordered_preds_ids)
```

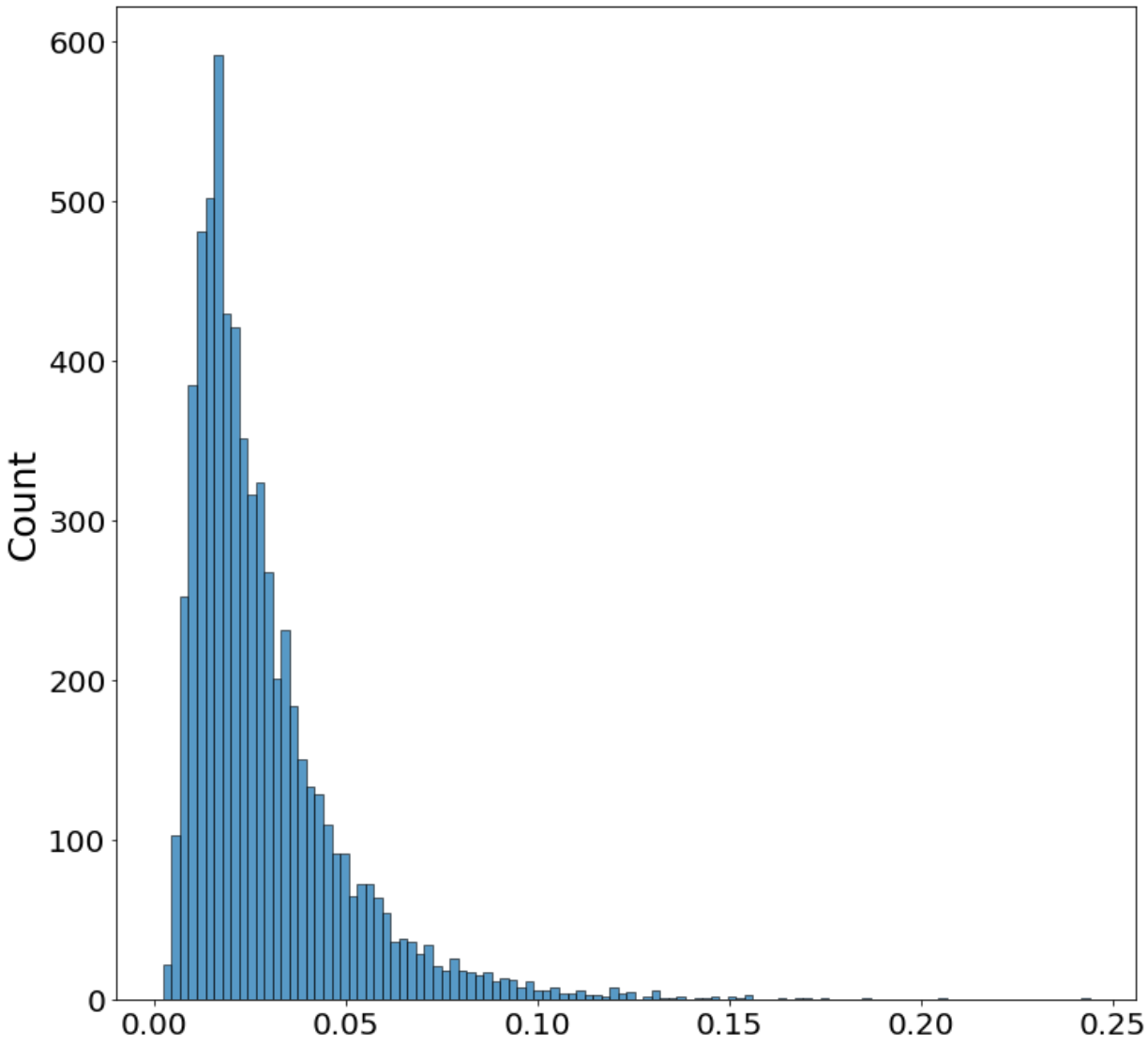
Out[13]:
```
999866
```

In [9]:
```
from tqdm import tqdm
import multiprocessing as mp
from ocpmodels.modules.evaluator import Evaluator

evaluator = Evaluator(task="s2ef")

def eval_wrapper(mp_arg):
    preds_id, predx, annox = mp_arg
    metrics = evaluator.eval(predx, annox, {})
    return preds_id, \
        metrics["forces_mae"]["metric"], \
        metrics["forces_cos"]["metric"], \
        metrics["forces_magnitude"]["metric"], \
        torch.abs(predx["forces"] - annox["forces"]).mean(dim=1), \
        torch.norm(predx["forces"], p=2, dim=1), \
        torch.norm(annox["forces"], p=2, dim=1),

mp_args = [
    (
        reordered_preds_ids[idx],
        {
            "energy": reordered_preds["energy"][idx],
            "forces": reordered_preds["forces"][idx],
            "natoms": reordered_preds["natoms"][idx],
        },
        {
            "energy": tensorized_annos["energy"][idx],
            "forces": tensorized_annos["forces"][idx],
            "natoms": tensorized_annos["natoms"][idx],
        },
    )
    for idx in tqdm(range(0, len(reordered_preds_ids), int(1000000 / 6500))
]

with mp.Pool(20) as p:
    sys_metrics = list(tqdm(p.imap(eval_wrapper, mp_args), total=len(mp_arg
```

```
100%|██████████| 6536/6536 [00:00<00:00, 221401.80it/s]
100%|██████████| 6536/6536 [00:14<00:00, 446.83it/s]
```

In [40]:
```
sns.histplot([k[1] for k in sys_metrics])
```

Out[40]:
```
<AxesSubplot:ylabel='Count'>
```

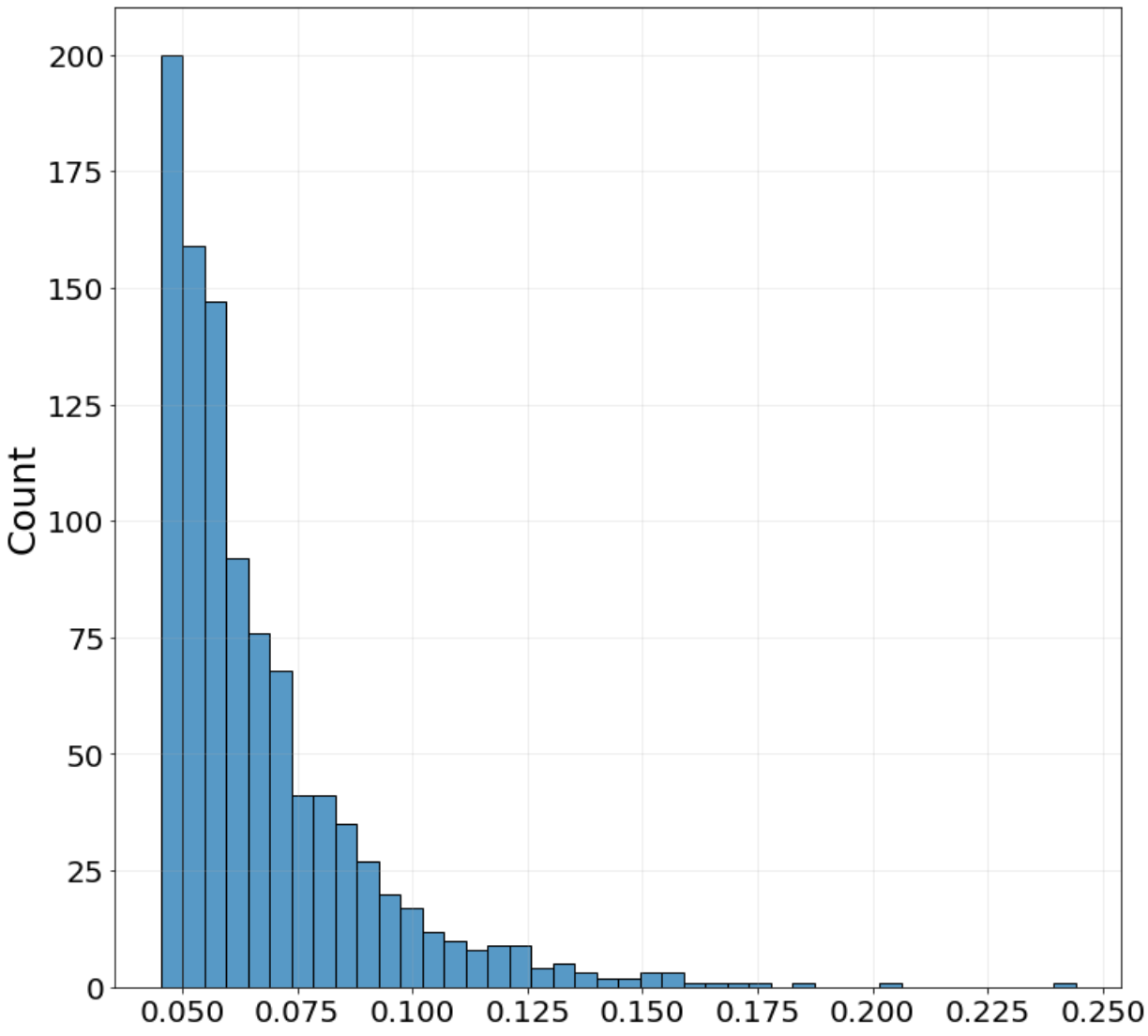


In [16]:
```
sm = sorted(sys_metrics, key=lambda x: -x[1]) # first index is force_mae
# sm = sorted(sys_metrics, key=lambda x: x[2]) # second index is force_cos
```

In [ ]:
```
sm[:100]
```

In [45]:
```
import matplotlib.ticker as ticker

smm = [k[1] for k in sm[:1000]]

ax = sns.histplot(smm)
ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
ax.grid(alpha=0.25)

print(np.mean(smm))
print(np.median(smm))
print(np.min(smm))
print(np.max(smm))
```

```
0.06714243528060149
0.05939543980454642
0.04535723200031355
0.24407549257631656
```



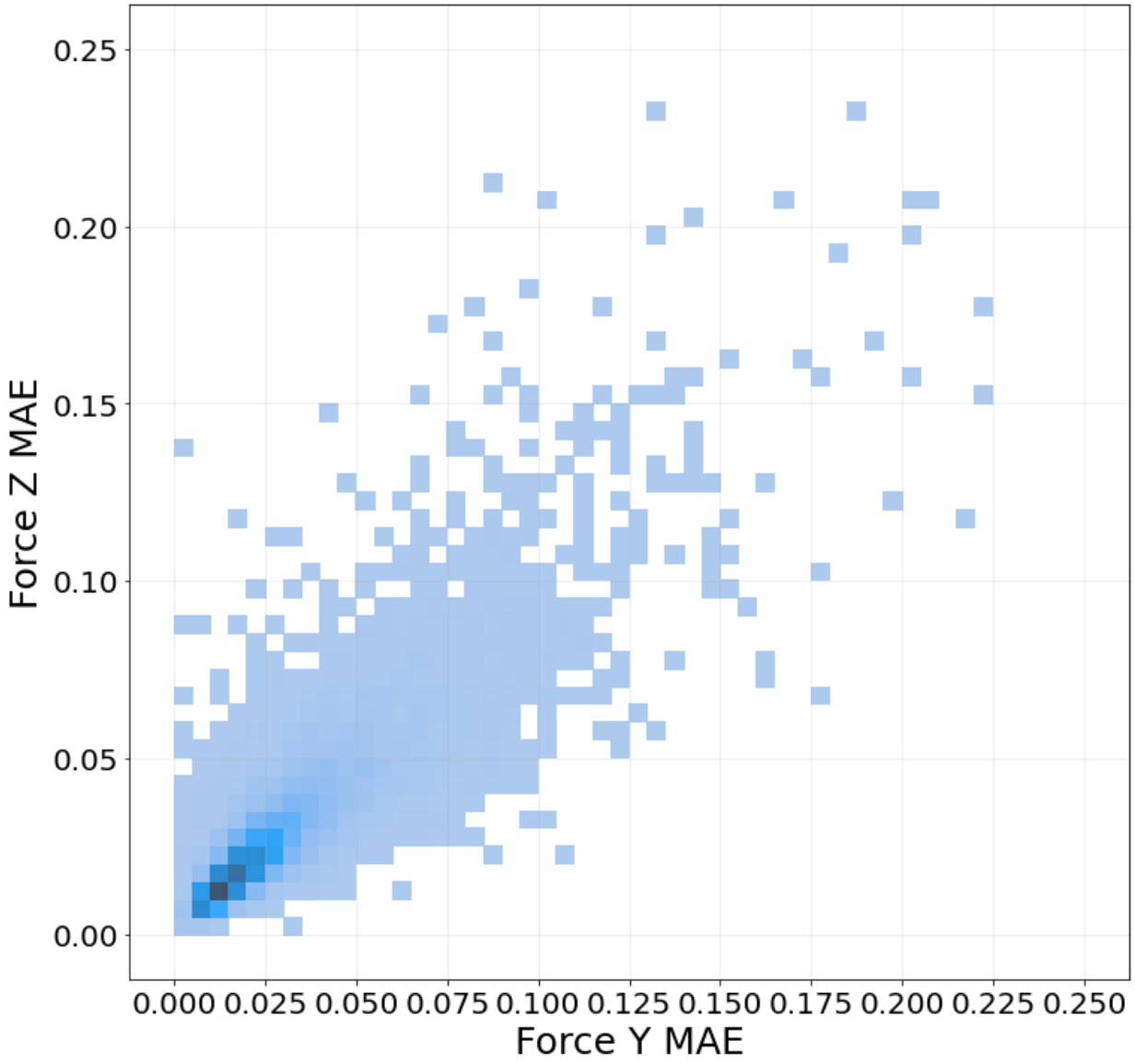
In [265]:
```
# create a dataframe with both MAE and cosine
df = pd.DataFrame(sm, columns=["id", "Force MAE", "Force cosine", "Force ma
df
```

Out[265]:

	id	Force MAE	Force cosine	Force magnitude	Force X MAE	Force Y MAE	Force Z MAE
0	1289072_2	0.209823	0.902933	0.242660	0.209683	0.187614	0.232173
1	2094144_8	0.203367	0.964596	0.261460	0.207951	0.204612	0.197537
2	722967_13	0.201065	0.925898	0.234435	0.192722	0.202078	0.208395
3	1224373_4	0.200547	0.873371	0.207685	0.186785	0.208469	0.206387
4	749103_61	0.192052	0.868234	0.241213	0.179139	0.220624	0.176394
...	...	...	...	...	...	...	...
9995	1690726_158	0.003591	0.428780	0.004365	0.002215	0.005541	0.003017
9996	431754_54	0.003572	0.898677	0.003981	0.001916	0.003586	0.005214
9997	1311004_258	0.003367	0.813950	0.005130	0.001828	0.003763	0.004509
9998	1289950_234	0.003073	0.147953	0.003059	0.002438	0.003199	0.003583
9999	1727238_162	0.002585	0.446942	0.002945	0.001848	0.003328	0.002580

10000 rows × 7 columns

In [268]:
```
# ax = sns.histplot(df, x="Force MAE", y="Force cosine", binrange=[[0.0, 0.
# ax = sns.histplot(df, x="Force MAE", y="Force magnitude", binrange=[[0.0,
# ax = sns.histplot(df, x="Force magnitude", y="Force cosine", binrange=[[0

ax = sns.histplot(df, x="Force Y MAE", y="Force Z MAE", binrange=[[0.0, 0.2
ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
ax.yaxis.set_major_locator(ticker.MultipleLocator(0.05))
ax.grid(alpha=0.25)
# plt.xlabel("Force magnitude error")
```



Let's look at individual samples with high force MAE -- all atoms / directions or individual atoms / directions

In [17]:
```
sm[0]
```

Out[17]:
```
('1558502_234',
 0.24407549257631656,
 0.7327561908298068,
 0.4400981532202827,
 tensor([0.2311, 0.4377, 0.2806, 0.4162, 0.2864, 0.1028, 0.1607, 0.0832,
0.1981]),
 tensor([0.3599, 0.1339, 0.0860, 0.6021, 0.4390, 0.1418, 0.0683, 0.1667,
0.3953],
        dtype=torch.float16),
 tensor([1.0287, 0.8144, 0.5613, 1.2671, 0.9357, 0.2649, 0.3491, 0.3591,
0.7735]))
```

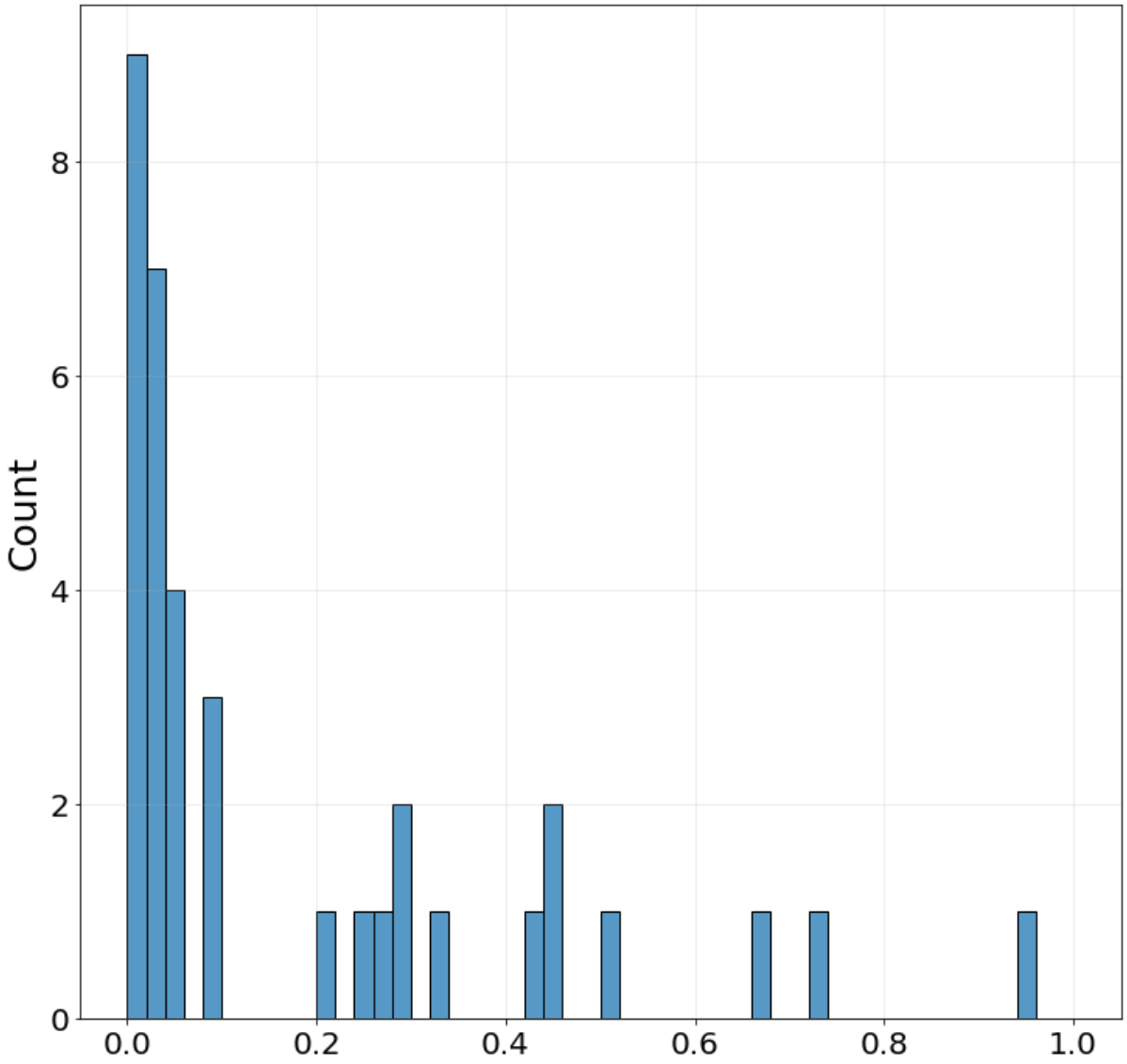
In [301]:
```
entr = sm[0]
print(entr)

idx = entr[0]

ind = reordered_preds_ids.tolist().index(idx)
assert reordered_preds_ids[ind] == annos_ids[ind]

fp = reordered_preds["forces"][ind]
fa = tensorized_annos["forces"][ind]
assert fp.shape == fa.shape

#
print("error separately across 3 dims")
err = torch.abs(fa - fp).mean(dim=0)
print(err)

#
err = torch.cosine_similarity(fa, fp)
print("cosine", err.mean(), err)

#
print("error across individual atoms")
err = torch.abs(fa - fp).mean(dim=1)
print(err.shape, err)

#
print("force anno norm")
err = torch.norm(fa, p=2, dim=1)
print(err.mean(), err)

#
print(fp)

#
print(fa)

err = torch.abs(fa - fp).mean(dim=1)
plt.grid(alpha=0.25)
sns.histplot(err, binrange=[0.0, 1.0], bins=50)
```

```
('1289072_2', 0.20982321318205413, 0.9029326567778716, 0.2426595430116395
6, 0.2096829285492768, 0.1876138352059029, 0.23217283712851033)
error separately across 3 dims
tensor([0.2097, 0.1876, 0.2322])
cosine tensor(0.9029) tensor([0.9748, 0.7980, 0.5794, 0.6054, 0.9963, 0.9
443, 0.9871, 0.9873, 0.7537,
        0.9655, 1.0000, 0.9932, 1.0001, 0.9863, 0.9981, 0.9928, 0.9998,
0.9640,
        0.9635, 0.9915, 0.9849, 0.9464, 0.9497, 1.0002, 0.5104, 1.0001,
0.9745,
        0.9989, 0.9920, 0.9988, 0.9926, 0.9968, 0.9973, 0.5335, 0.5268,
0.9950,
        0.5293])
error across individual atoms
torch.Size([37]) tensor([4.5127e-02, 2.0987e-01, 5.2254e-02, 9.4136e-01,
1.6680e-02, 2.1045e-02,
```

```
        2.5078e-02, 2.4023e-02, 4.4635e-01, 8.0766e-02, 3.0552e-03, 8.721
8e-02,
        1.0185e-03, 3.1003e-02, 9.1654e-03, 3.6427e-02, 3.2280e-03, 2.846
1e-02,
        5.5781e-02, 4.7265e-02, 2.6265e-02, 8.7756e-02, 2.8162e-01, 3.635
6e-03,
        3.2043e-01, 1.8878e-03, 4.4740e-01, 1.1546e-02, 2.4701e-01, 7.752
1e-03,
        2.9200e-01, 2.6584e-01, 4.3005e-01, 5.0515e-01, 1.2784e+00, 7.223
0e-01,
        6.6927e-01])
force anno norm
tensor(1.2617) tensor([0.2126, 0.5689, 0.1445, 2.1180, 0.1714, 0.1830, 0.
1645, 0.1630, 1.1599,
        0.5098, 0.6296, 0.8432, 0.6027, 0.4134, 0.3528, 0.3277, 0.3496,
0.1829,
        0.3902, 0.1684, 0.1047, 0.5036, 1.2982, 0.6483, 0.8452, 0.6252,
3.2866,
        0.3695, 3.7938, 0.3540, 2.9804, 7.1839, 2.2448, 1.1986, 2.6413,
7.7916,
        1.1572])
tensor([[-4.6425e-03, -4.7455e-02, -1.1902e-01],
        [-5.7190e-02, -9.2834e-02, -2.0093e-01],
        [ 9.3307e-03,  1.0437e-01, -1.3763e-02],
        [ 1.5459e+00, -1.3242e+00,  6.0693e-01],
        [-4.9973e-03, -1.1536e-01, -1.6333e-01],
        [ 2.8839e-03, -4.1595e-02,  1.8201e-01],
        [-3.7994e-02,  8.5510e-02,  8.2764e-02],
        [ 3.5461e-02,  8.8196e-02,  8.2031e-02],
        [-1.2510e+00, -6.4746e-01,  2.5659e-01],
        [-1.1200e-01,  3.1665e-01,  1.4600e-01],
        [ 8.1635e-04, -5.2979e-01,  3.3008e-01],
        [-3.2910e-01, -4.7461e-01,  3.4424e-01],
        [-4.0817e-04,  5.5469e-01, -2.3804e-01],
        [ 1.1879e-02,  3.7354e-01,  2.0776e-01],
        [-9.2697e-03,  3.4155e-01, -9.8999e-02],
        [ 4.0924e-02, -3.7012e-01, -1.1517e-01],
        [ 4.7684e-03, -2.8882e-01,  1.8530e-01],
        [-5.0323e-02, -1.3220e-01, -1.2195e-01],
        [-2.7634e-02, -1.6956e-01,  2.6001e-01],
        [-2.3026e-02,  6.5369e-02,  1.6113e-02],
        [ 8.2245e-03,  4.7668e-02,  2.4994e-02],
        [ 3.9307e-01,  3.6963e-01,  7.8064e-02],
        [-1.2432e+00,  5.1318e-01, -7.3926e-01],
        [-2.3232e-03, -5.4541e-01,  3.4131e-01],
        [-6.1371e-02,  3.1030e-01,  5.5237e-02],
        [ 9.3758e-05,  5.7373e-01, -2.4756e-01],
        [ 1.0391e+00,  2.1680e+00,  9.7900e-01],
        [ 9.8419e-03,  3.3789e-01, -8.3313e-02],
        [ 1.1729e+00, -2.3867e+00, -2.1035e+00],
        [-7.1259e-03, -3.0762e-01,  1.9580e-01],
        [ 6.6504e-01, -1.7188e+00,  1.7646e+00],
        [ 9.5508e-01, -1.3477e-01,  7.1406e+00],
        [-2.4395e+00,  1.6074e+00, -7.9297e-01],
        [-6.0791e-01, -6.5552e-02,  5.5078e-01],
        [-2.1362e-01, -6.8213e-01,  4.6265e-01],
        [ 1.8359e+00,  1.4805e+00, -5.3320e+00],
```

```
                [ 7.6416e-01, -7.4036e-02,  1.1172e+00]], dtype=torch.float16)
        tensor([[ 1.7970e-03, -1.1977e-01, -1.7564e-01],
                [ 4.8340e-02,  6.9617e-02, -5.6255e-01],
                [ 1.4787e-02,  6.6670e-02, -1.2737e-01],
                [ 2.0820e+00,  3.8776e-01,  3.0886e-02],
                [ 6.9136e-03, -1.0645e-01, -1.3411e-01],
                [ 1.8511e-03,  2.0314e-02,  1.8181e-01],
                [-2.4528e-02,  1.1513e-01,  1.1491e-01],
                [ 2.1605e-02,  1.1311e-01,  1.1533e-01],
                [-5.0275e-01, -6.3232e-01,  8.3227e-01],
                [-1.0814e-01,  3.8581e-01,  3.1528e-01],
                [-1.0487e-03, -5.3463e-01,  3.3253e-01],
                [-3.2756e-01, -6.4334e-01,  4.3562e-01],
                [ 1.2442e-03,  5.5357e-01, -2.3832e-01],
                [-5.6567e-02,  3.5824e-01,  1.9850e-01],
                [-8.1683e-03,  3.4450e-01, -7.5549e-02],
                [ 1.0593e-02, -3.0226e-01, -1.2627e-01],
                [ 6.3343e-03, -2.9661e-01,  1.8498e-01],
                [-1.6414e-02, -1.0657e-01, -1.4779e-01],
                [ 1.8279e-02, -2.8270e-01,  2.6830e-01],
                [-3.8112e-02,  1.5526e-01,  5.2931e-02],
                [-2.3736e-03,  9.3199e-02,  4.7661e-02],
                [ 2.8541e-01,  3.5334e-01,  2.1739e-01],
                [-9.0822e-01,  1.7502e-01, -9.1102e-01],
                [ 1.9152e-03, -5.4961e-01,  3.4378e-01],
                [-7.6755e-01,  2.5018e-01,  2.5023e-01],
                [-1.2878e-03,  5.7269e-01, -2.5080e-01],
                [ 9.4658e-01,  2.5450e+00,  1.8517e+00],
                [ 8.6767e-03,  3.6189e-01, -7.3841e-02],
                [ 1.2785e+00, -2.3435e+00, -2.6957e+00],
                [-1.0188e-02, -3.0676e-01,  1.7646e-01],
                [ 1.1175e+00, -1.9816e+00,  1.9253e+00],
                [ 5.0323e-01,  1.8771e-01,  7.1638e+00],
                [-1.7520e+00,  1.3193e+00, -4.7837e-01],
                [-1.3657e-01, -9.7089e-01,  6.8955e-01],
                [-1.9097e+00,  9.7414e-02,  1.8221e+00],
                [ 1.9226e+00,  1.4906e+00, -7.4021e+00],
                [-1.3367e-01, -8.4554e-01,  7.7871e-01]])
```

Out[301]: <AxesSubplot:ylabel='Count'>



In [83]:
```
# Let's look at the absolute error per atom for the first 1k worst force MA
# How many atoms are in 1k structures?
smm = sm[:1000]

print("atoms:", sum([len(k[4]) for k in smm]))
err_and_norm = list(zip(
    torch.cat([k[4] for k in smm]).tolist(), torch.cat([k[6] for k in smm])
))
print(len(err_and_norm))
```

```
atoms: 23581
23581
```

In [84]:
```
# create a dataframe with both absolute error per atom and true force norm
df = pd.DataFrame(err_and_norm, columns=["Force absolute error", "GT force
df
```

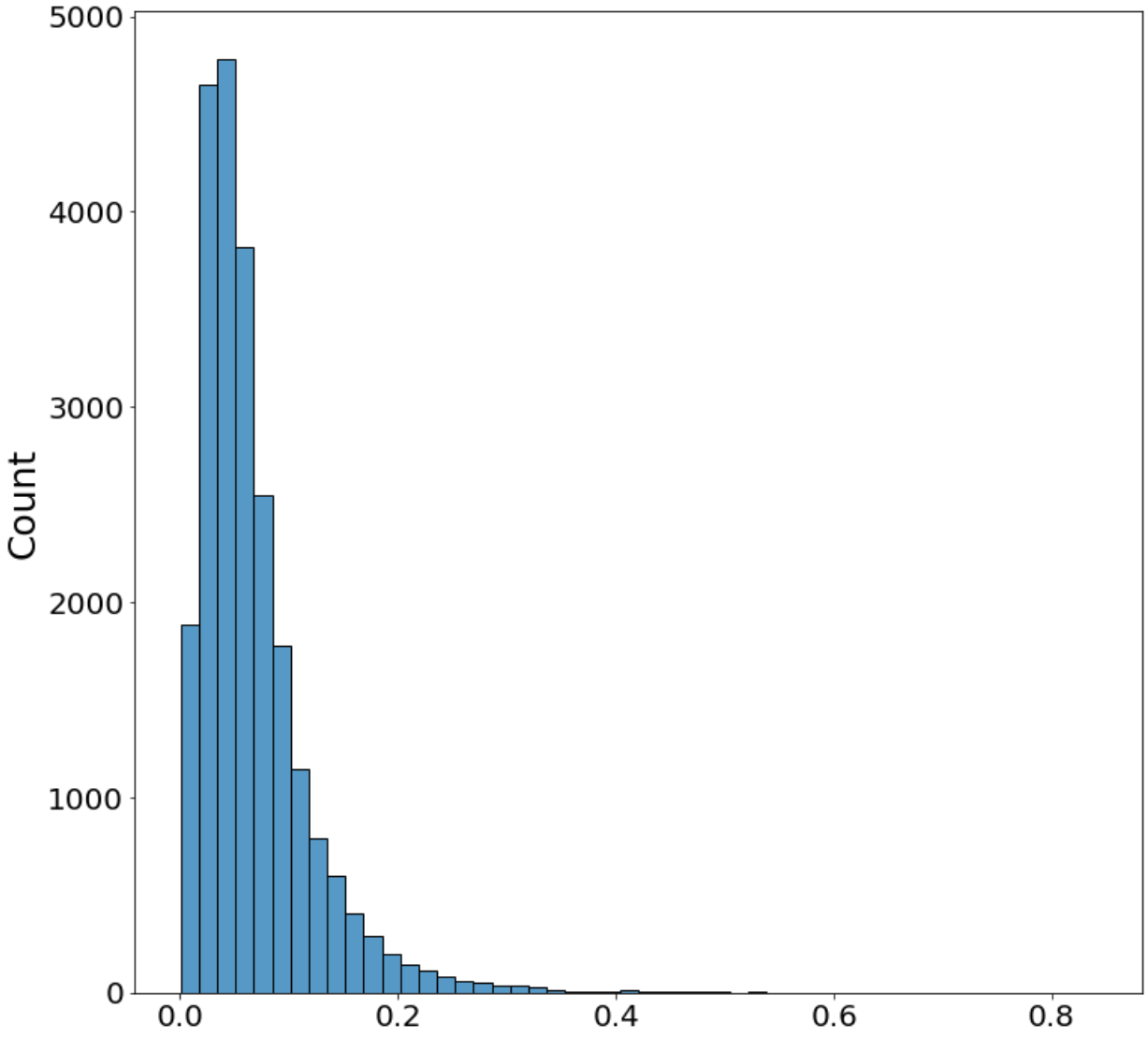
Out[84]:

	Force absolute error	GT force norm
0	0.231112	1.028724
1	0.437695	0.814389
2	0.280570	0.561283
3	0.416209	1.267113
4	0.286365	0.935717
...	...	...
23576	0.058692	2.306966
23577	0.017444	1.746910
23578	0.035169	1.352924
23579	0.039764	0.548216
23580	0.067856	3.282638

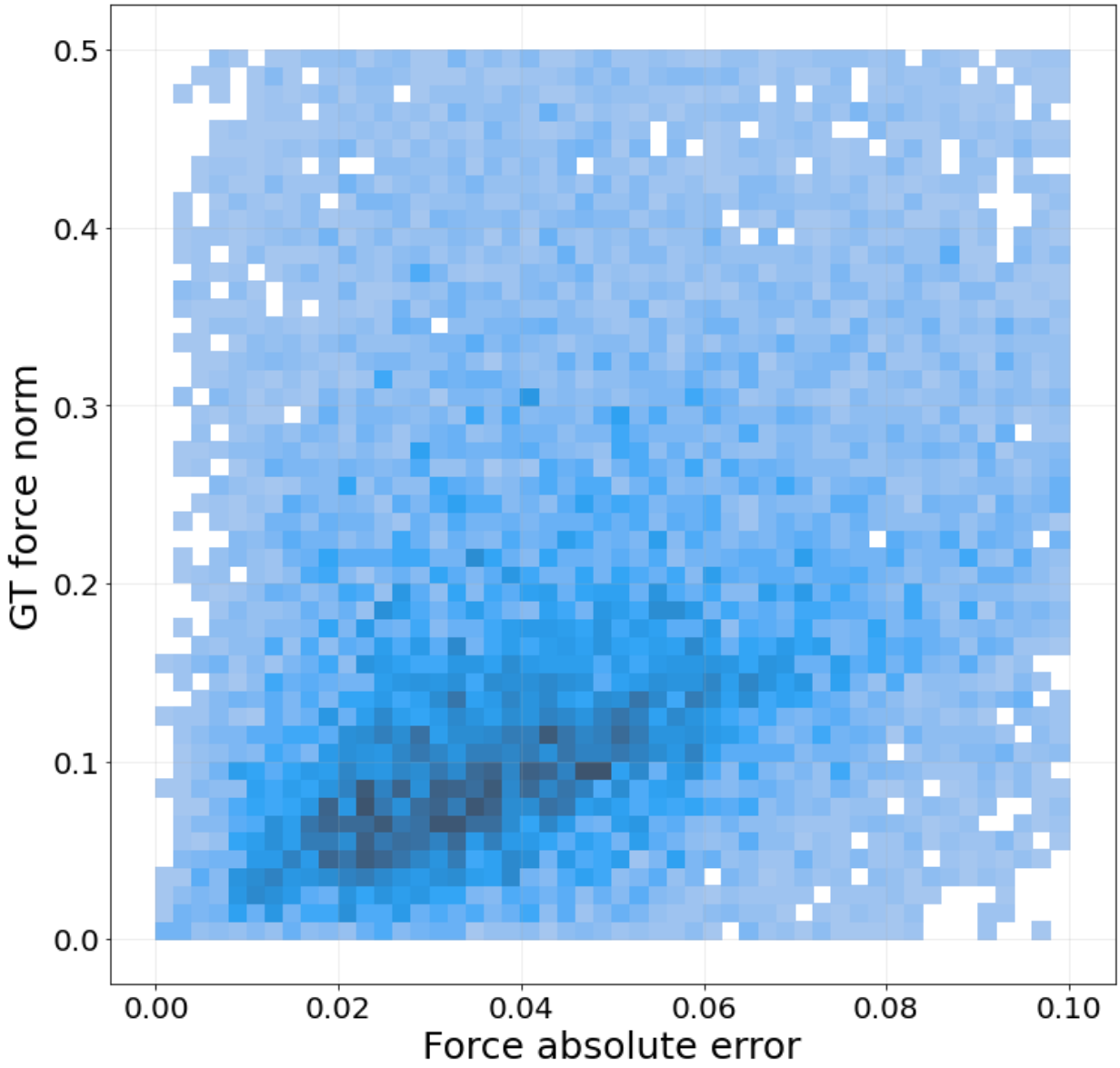
23581 rows × 2 columns

In [86]:
```
smmm = [k[0] for k in err_and_norm]

ax = sns.histplot(smmm, bins=50)
# ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
# ax.grid(alpha=0.25)

print(np.mean(smmm))
print(np.median(smmm))
print(np.min(smmm))
print(np.max(smmm))
```

```
0.06675726290576602
0.05241391435265541
0.00021147902589291334
0.8407263159751892
```



In [88]:
```
# ax = sns.histplot(df, x="Force absolute error", y="GT force norm")
ax = sns.histplot(df, x="Force absolute error", y="GT force norm", binrange
# ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
# ax.yaxis.set_major_locator(ticker.MultipleLocator(0.05))
ax.grid(alpha=0.25)
```



Extract 10k atoms with worst force absolute error

In [91]:
```
err_and_norm = list(zip(
    torch.cat([k[4] for k in sm]).tolist(), torch.cat([k[6] for k in sm]).t
))
print(len(err_and_norm))

err_and_norm = sorted(err_and_norm, key=lambda x: -x[0])
print(err_and_norm[:10])
```

```
161546
[(0.8407263159751892, 2.330404758453369), (0.8020208477973938, 3.24352717
39959717), (0.7101323008537292, 5.4727935791015625), (0.6571912169456482,
1.3692362308502197), (0.6560153365135193, 4.823131561279297), (0.64982146
02470398, 0.6692764759063721), (0.5654703378677368, 2.254974365234375),
(0.5417746901512146, 5.925816535949707), (0.5323588848114014, 0.360300809
14497375), (0.52813321352005, 1.1368290185928345)]
```

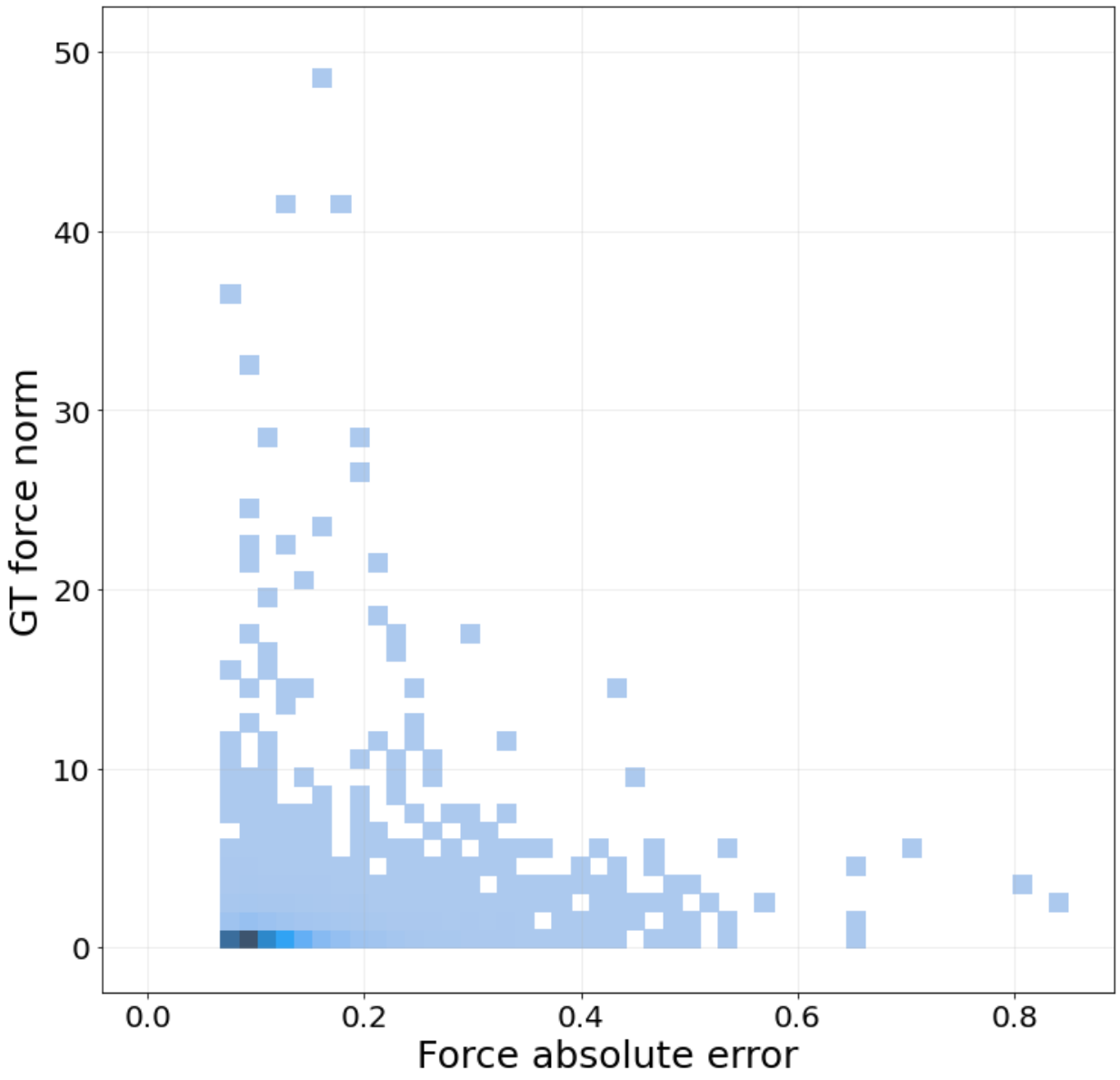
In [120]:
```
df = pd.DataFrame(err_and_norm[:10000], columns=["Force absolute error", "G
df
```

Out[120]:

	Force absolute error	GT force norm
0	0.840726	2.330405
1	0.802021	3.243527
2	0.710132	5.472794
3	0.657191	1.369236
4	0.656015	4.823132
...	...	...
9995	0.076146	0.224911
9996	0.076134	0.418604
9997	0.076131	0.265097
9998	0.076126	0.500314
9999	0.076124	0.400850

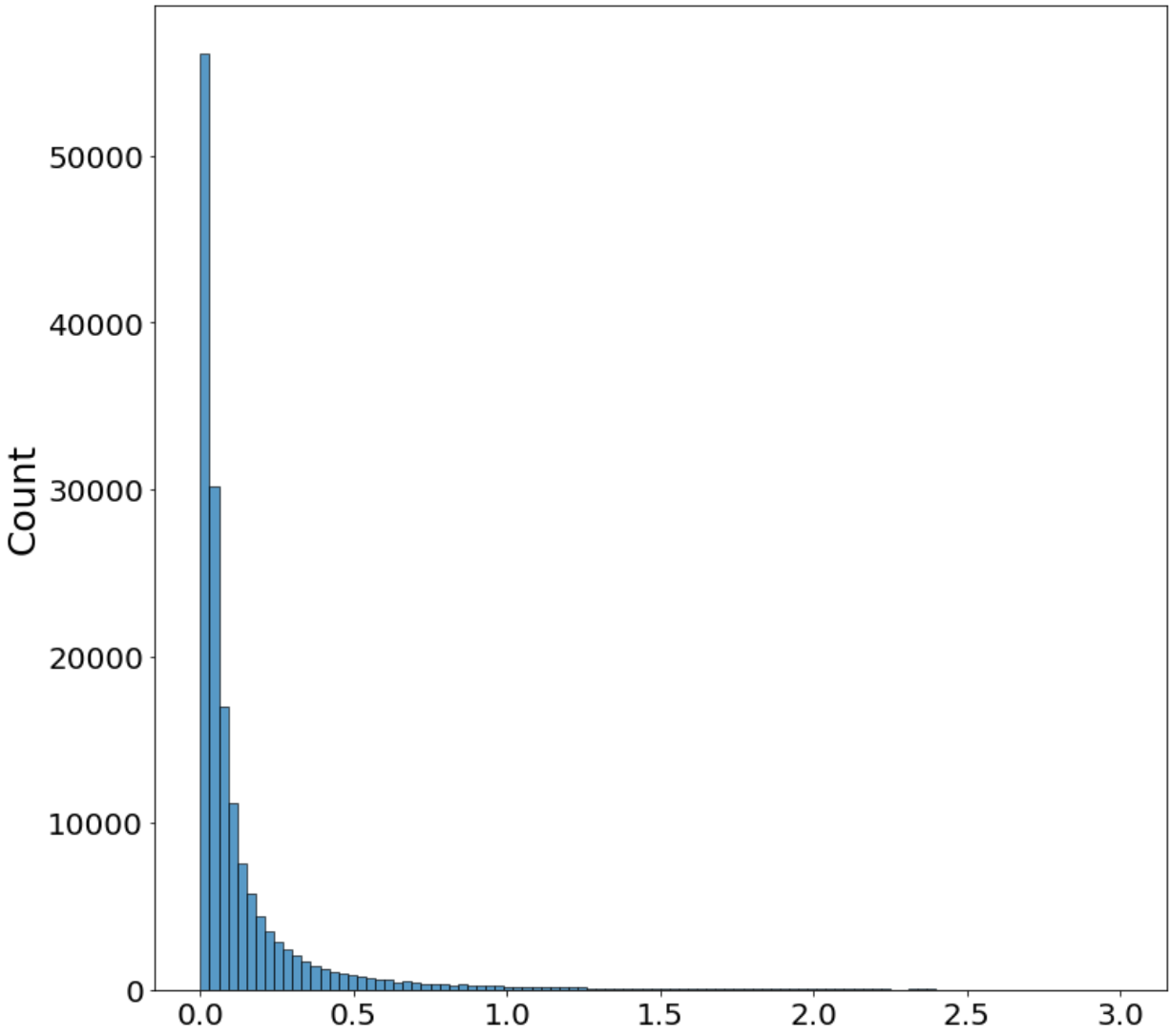
10000 rows × 2 columns

In [121]:
```
# ax = sns.histplot(df, x="Force absolute error", y="GT force norm")
ax = sns.histplot(
    df,
    x="Force absolute error",
    y="GT force norm",
    binrange=[[0, 0.85],[0, 50]],
    bins=50
)
# ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
# ax.yaxis.set_major_locator(ticker.MultipleLocator(0.05))
ax.grid(alpha=0.25)
```



In [117]:
```
smmm = [k[1] for k in err_and_norm]

sns.histplot(smmm, bins=100, binrange=[0, 3.0])

print(np.mean(smmm))
print(np.median(smmm))
print(np.min(smmm))
print(np.max(smmm))
```

```
0.16888592987248424
0.052861614152789116
0.0001138657535193488
48.544654846191406
```



Read individual trajectories and visualize force annotations / ground-truth

In [123]:
```
sm = sorted(sys_metrics, key=lambda x: -x[1]) # first index is force_mae
# sm = sorted(sys_metrics, key=lambda x: x[2]) # second index is force_cos
```

In [124]:
```
print(sm[0])
```

```
('1558502_234', 0.24407549257631656, 0.7327561908298068, 0.4400981532202827, tensor([0.2311, 0.4377, 0.2806, 0.4162, 0.2864, 0.1028, 0.1607, 0.0832, 0.1981]), tensor([0.3599, 0.1339, 0.0860, 0.6021, 0.4390, 0.1418, 0.0683, 0.1667, 0.3953],
       dtype=torch.float16), tensor([1.0287, 0.8144, 0.5613, 1.2671, 0.9357, 0.2649, 0.3491, 0.3591, 0.7735]))
```

In [160]:
```
import ase
from ase.io import read, write
#
from ase.gui.gui import GUI
from ase.gui.save import save_dialog
from PIL.Image import open
from io import BytesIO
import math

def traj_to_image_with_force(atoms, filename="test.png", rotations="", scal
    """ Accept 1 atoms object and output to png
        uses ase.gui Tkinter interface (hopefully no need to really ask a d
        PIL is used to convert eps to png
    """
    gui = GUI(images=[atoms], rotations=rotations)
    gui.scale *= scale
    gui.window['toggle-show-forces'] = True
    gui.draw()
    # Save current canvas to EPS format.
    eps_string = gui.window.canvas.postscript()
    with open(BytesIO(eps_string.encode("ascii")),
              formats=["EPS"]) as image:
        # PDF to raster dpi conversion
        # see https://stackoverflow.com/questions/28344258/with-pythons-pil
        image.load(math.ceil(dpi / 72.0))
        image.save(filename)
    gui.exit()
```

In [152]:
```
idx = 1

print(sm[idx])

traj_id = sm[idx][0].split("_")[0]
traj_frames = ase.io.read(
    os.path.join(
        "/checkpoint/electrocatalysis/relaxations/bulkadsorbate/restitch_ja
        "random" + traj_id + ".traj"
    ),
    ":"
)
print("length of traj", len(traj_frames))

frame_id = int(sm[idx][0].split("_")[1])
frame = traj_frames[frame_id]
print(type(frame))
```

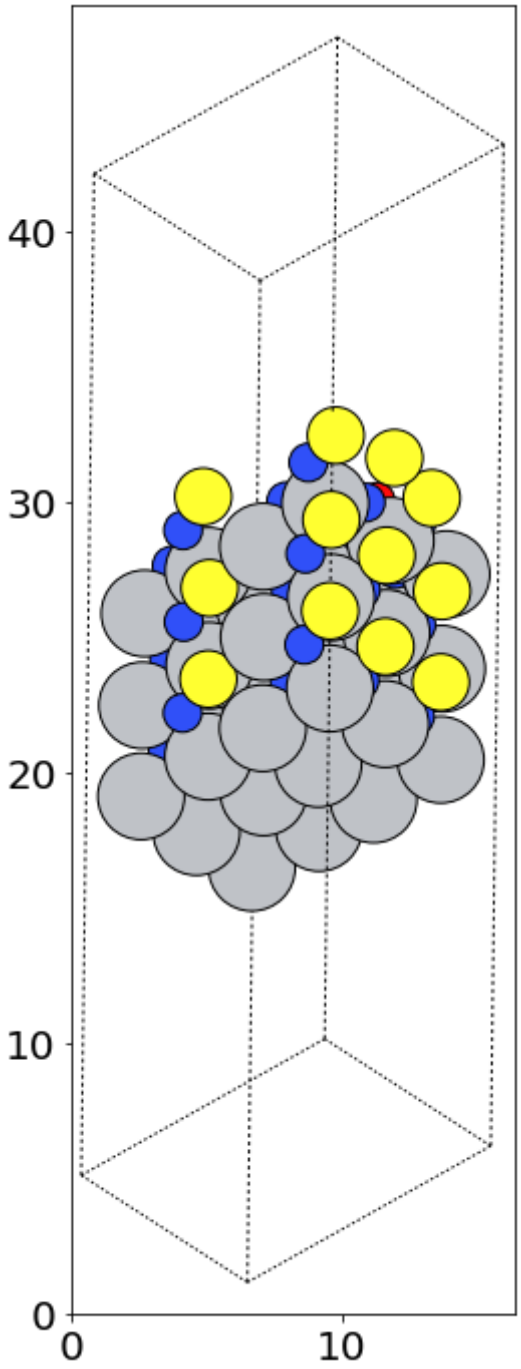
```
('1608794_46', 0.20514368543437883, 0.725238182965447, 0.212003792033476,
tensor([0.0769, 0.1805, 0.2076, 0.1950, 0.1653, 0.2494, 0.0832, 0.3134,
0.0766,
        0.3460, 0.0334, 0.2262, 0.1939, 0.1777, 0.3043, 0.2162, 0.1348,
0.4891,
        0.1489, 0.1399, 0.1534, 0.3232, 0.1199, 0.2225, 0.1067, 0.2527,
0.1621,
        0.2317, 0.2727, 0.1873, 0.4742, 0.3463, 0.0421, 0.1218]), tensor
([0.2605, 0.2166, 0.3738, 0.4119, 1.0391, 0.6377, 0.5806, 0.1790, 0.7310,
        1.0488, 0.0965, 0.4800, 0.9355, 0.3960, 1.0010, 0.4260, 0.5986,
0.4976,
        0.3425, 0.1785, 0.4602, 0.2144, 1.0625, 3.0215, 0.1971, 0.7686,
1.6768,
        1.9775, 0.3511, 0.5806, 0.7642, 0.8291, 1.3711, 1.4971],
       dtype=torch.float16), tensor([0.4101, 0.4100, 0.5754, 0.4452, 1.20
25, 0.8027, 0.4659, 0.5048, 0.7882,
        1.6175, 0.1282, 0.8556, 1.1911, 0.5921, 1.4013, 0.7718, 0.7987,
0.5078,
        0.5378, 0.2032, 0.5686, 0.7008, 1.0106, 3.2797, 0.3941, 1.2632,
1.5486,
        2.1692, 0.5406, 0.3632, 0.5877, 0.3210, 1.4224, 1.3564]))
length of traj 568
<class 'ase.atoms.Atoms'>
```

In [153]:
```
frame.constraints
```

Out[153]:
```
[FixAtoms(indices=[0, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 15, 18, 19, 20, 2
1, 24, 25, 26, 27, 30, 31, 32, 33, 36, 37, 39, 40, 42, 43, 45, 46, 48, 4
9, 51, 52, 54, 55, 56, 57, 60, 61, 62, 63, 66, 67, 68, 69, 72, 73, 74, 7
5, 78, 79, 80, 81, 84, 85, 86, 87])]
```

In [167]:
```
fig, ax = plt.subplots(1)
ase.visualize.plot.plot_atoms(frame, ax, scale=1.0, rotation=("-75x, 45y, 1
```

Out[167]: <AxesSubplot:>



Plot error as a function of where in traj frame is from

In [233]:
```
sm = sorted(sys_metrics, key=lambda x: -x[1]) # first index is force_mae

sid_fid_atomid = []
for syst in sm:
    for i in range(len(syst[4])):
        sid_fid_atomid.append(syst[0])
print(len(sid_fid_atomid))
```

```
161546
```

In [234]:
```
err_and_norm = list(zip(
    sid_fid_atomid, torch.cat([k[4] for k in sm]).tolist(), torch.cat([k[6]
))
print(len(err_and_norm))
```

```
161546
```

In [235]:
```
err_and_norm = [list(k) for k in err_and_norm]
```

In [236]:
```
err_and_norm[0]
```

Out[236]:
```
['1558502_234', 0.23111212253570557, 1.0287237167358398]
```

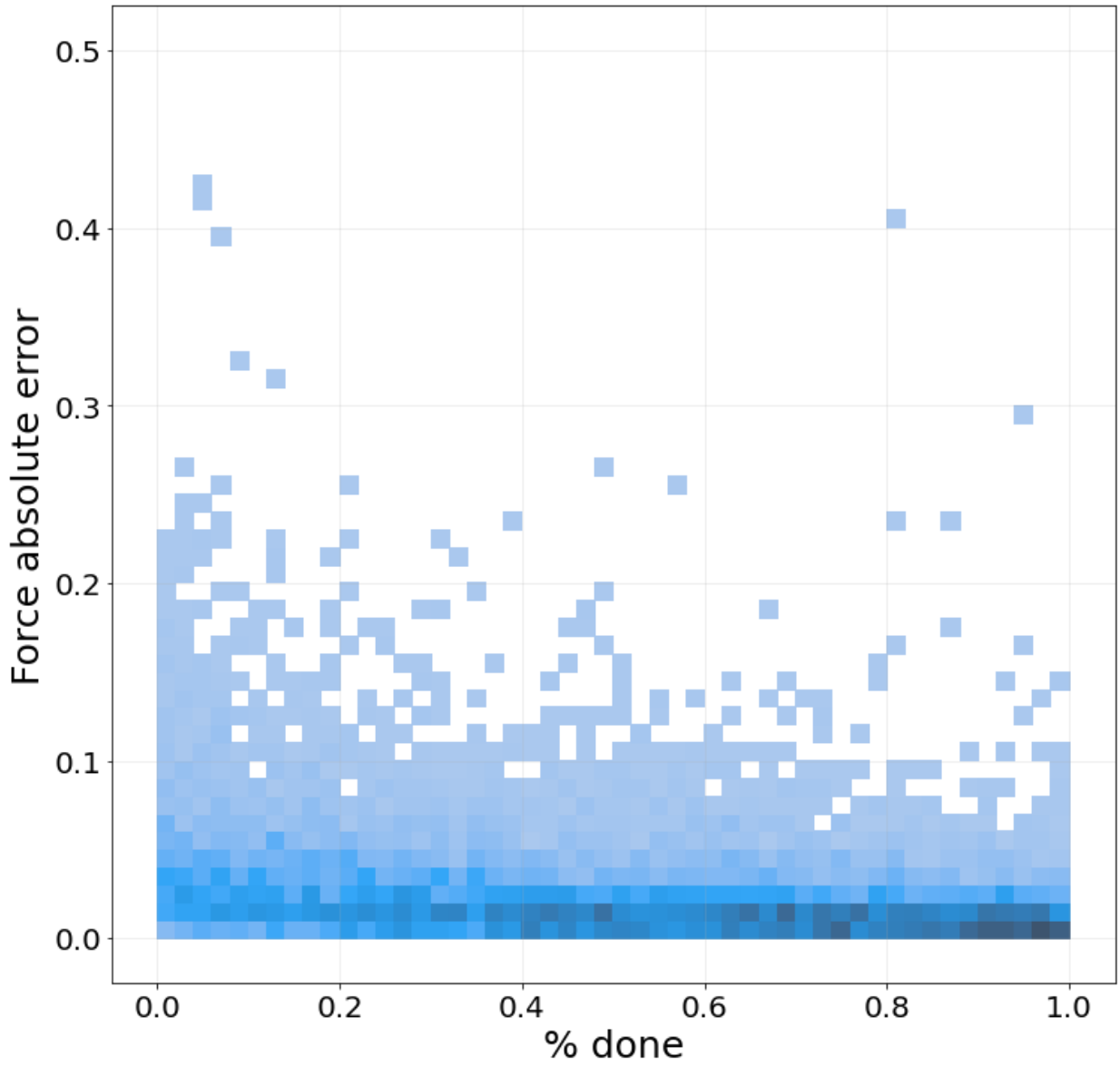
In [237]:
```
from ase.io import Trajectory

idx = []

for i in tqdm(range(0, 161546, 15)):
    # read traj, frame
    traj_id = err_and_norm[i][0].split("_")[0]
    traj_frames = Trajectory(os.path.join(
        "/checkpoint/electrocatalysis/relaxations/bulkadsorbate/restitch_ja
        "random" + traj_id + ".traj"
    ))
    # get % done
    frame_id = int(err_and_norm[i][0].split("_")[1])
    frame = traj_frames[frame_id]
    done = frame_id / len(traj_frames)
    # save to err_and_norm
    err_and_norm[i].append(frame_id)
    err_and_norm[i].append(done)
    idx.append(i)
```

```
100%|██████████| 10770/10770 [00:40<00:00, 264.88it/s]
```

In [238]:
```
[err_and_norm[k] for k in idx[:3]]
```

Out[238]:
```
[['1558502_234',
  0.23111212253570557,
  1.0287237167358398,
  234,
  0.8666666666666667],
 ['1608794_46',
  0.08323698490858078,
  0.46589237451553345,
  46,
  0.08098591549295775],
 ['1608794_46',
  0.3232371509075165,
  0.7007959485054016,
  46,
  0.08098591549295775]]
```

In [269]:
```
df = pd.DataFrame([err_and_norm[k] for k in idx], columns=["sid_fid", "Forc
df

# ax = sns.histplot(df, x="Force absolute error", y="GT force norm")
ax = sns.histplot(
    df,
    x="% done",
#     y="GT force norm",
    y="Force absolute error",
    binrange=[[0, 1.0],[0, 0.5]],
    bins=50,
)
# ax.xaxis.set_major_locator(ticker.MultipleLocator(0.025))
# ax.yaxis.set_major_locator(ticker.MultipleLocator(0.05))
ax.grid(alpha=0.25)
```



Errors per element

In [303]:
```
sm = sorted(sys_metrics, key=lambda x: -x[1]) # first index is force_mae

sid_fid_atomid = []
for syst in sm:
    for i in range(len(syst[4])):
        sid_fid_atomid.append(syst[0] + ".%d" % i)
print(len(sid_fid_atomid))
print(sid_fid_atomid[0])
```

```
161546
1558502_234.0
```

In [304]:
```
err_and_norm = list(zip(
    sid_fid_atomid, torch.cat([k[4] for k in sm]).tolist(), torch.cat([k[6]
))
print(len(err_and_norm))
```

```
161546
```

In [305]:
```
err_and_norm = [list(k) for k in err_and_norm]
```

In [306]:
```
from ase.io import Trajectory

idx = []

for i in tqdm(range(0, 161546)):
    # read traj, frame
    traj_id = err_and_norm[i][0].split("_")[0]
    traj_frames = Trajectory(os.path.join(
        "/checkpoint/electrocatalysis/relaxations/bulkadsorbate/restitch_ja
        "random" + traj_id + ".traj"
    ))
    # get atomic number
    frame_id = int(err_and_norm[i][0].split("_")[1].split(".")[0])
    frame = traj_frames[frame_id]
    natoms = frame.get_positions().shape[0]
    fixed_idx = np.zeros(natoms)
    fixed_idx[frame.constraints[0].index] = 1

    atomid = int(err_and_norm[i][0].split("_")[1].split(".")[1])
    z = frame.get_atomic_numbers()[fixed_idx == 0][atomid]
    # save to err_and_norm
    err_and_norm[i].append(z)
    idx.append(i)
```

```
100%|██████████| 161546/161546 [05:19<00:00, 505.00it/s]
```

In [307]:
```
natoms = traj_frames[0].get_positions().shape[0]
print(natoms)
fixed_idx = np.zeros(natoms)
fixed_idx[traj_frames[0].constraints[0].index] = 1
print(fixed_idx)

print(traj_frames[0].get_atomic_numbers())
```

```
38
[1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
 19 19 19 19 19 19 19 19  6  1  1  1  1  8]
```

In [308]:
```
z_to_err = {}
for i in tqdm(range(len(err_and_norm))):
    z = err_and_norm[i][-1]
    if z not in z_to_err:
        z_to_err[z] = []
    z_to_err[z].append(err_and_norm[i][1])
```

```
100%|██████████| 161546/161546 [00:00<00:00, 1607047.56it/s]
```

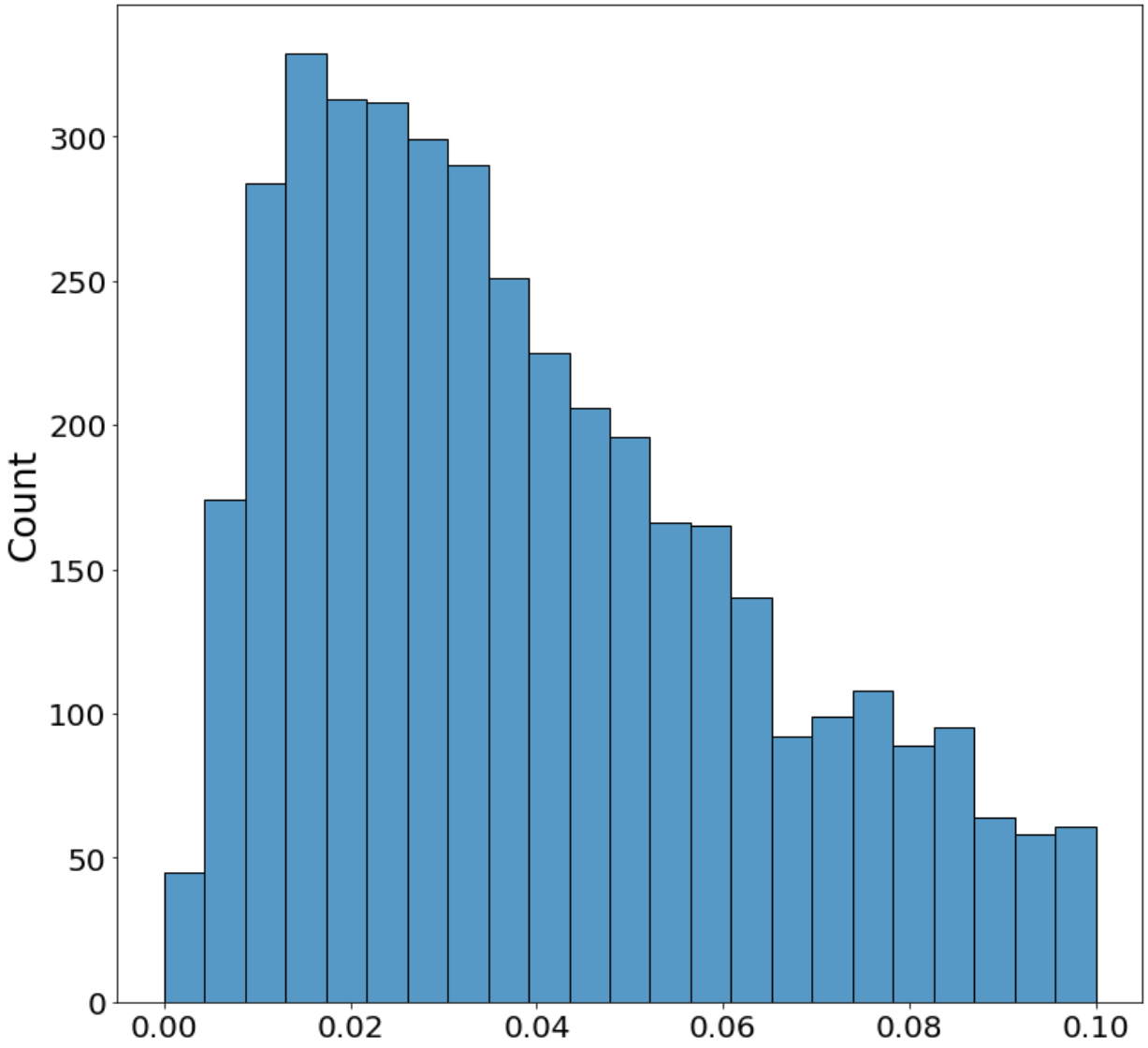
In [321]:
```
# print("Z", "cnt", "MAE", "Median AE", "Min AE", "Max AE")
for z in range(100):
    if z in z_to_err:
        print("%2d %5d %.04f %.04f %.04f %.04f" %
              (z, len(z_to_err[z]), np.mean(z_to_err[z]), np.median(z_to_er
#         if np.mean(z_to_err[z]) > 0.03:
#             print("YES")
```

```
 1 19109 0.0170 0.0114 0.0002 0.4969
 5    32 0.1473 0.1505 0.0442 0.3277
 6  9417 0.0480 0.0351 0.0009 0.5262
 7  4763 0.0588 0.0408 0.0005 0.8407
 8  6915 0.0447 0.0330 0.0009 0.6572
11  2600 0.0106 0.0082 0.0004 0.1404
13  4575 0.0246 0.0179 0.0004 0.2407
14  3917 0.0323 0.0223 0.0006 0.3695
15  3425 0.0346 0.0235 0.0004 0.3101
16  7740 0.0311 0.0218 0.0006 0.4891
17  2902 0.0226 0.0165 0.0005 0.2707
19  2335 0.0124 0.0104 0.0002 0.0640
20  3188 0.0182 0.0138 0.0003 0.1687
21  1901 0.0234 0.0167 0.0008 0.1615
22  3687 0.0293 0.0203 0.0006 0.4051
23  2018 0.0346 0.0229 0.0013 0.4564
24  1390 0.0389 0.0239 0.0010 0.3341
25  1189 0.0441 0.0287 0.0008 0.4377
26  1280 0.0351 0.0233 0.0007 0.4842
27  1667 0.0257 0.0179 0.0006 0.2782
28  2822 0.0215 0.0152 0.0004 0.2001
29  2554 0.0159 0.0114 0.0005 0.1205
30  2112 0.0193 0.0141 0.0002 0.2510
31  3512 0.0240 0.0181 0.0005 0.1936
32  2496 0.0266 0.0194 0.0004 0.3541
33  3055 0.0317 0.0223 0.0010 0.4964
34  5241 0.0299 0.0222 0.0006 0.3407
37  1385 0.0127 0.0101 0.0000 0.0882
38  2013 0.0204 0.0153 0.0002 0.1505
39  2281 0.0246 0.0183 0.0007 0.2458
40  2652 0.0295 0.0202 0.0004 0.2288
41  1751 0.0353 0.0232 0.0008 0.4377
42  1753 0.0382 0.0232 0.0003 0.3867
43   691 0.0384 0.0259 0.0018 0.4261
44  1448 0.0293 0.0192 0.0010 0.3015
45  2132 0.0271 0.0189 0.0011 0.2132
46  2898 0.0180 0.0132 0.0005 0.1617
47  2308 0.0152 0.0119 0.0009 0.1083
48  1415 0.0153 0.0115 0.0002 0.1347
49  2621 0.0211 0.0162 0.0008 0.1686
50  2591 0.0265 0.0207 0.0007 0.4424
51  2122 0.0305 0.0236 0.0005 0.3229
52  4721 0.0303 0.0220 0.0006 0.3800
55   901 0.0160 0.0134 0.0005 0.0879
72  3053 0.0306 0.0221 0.0008 0.2483
73  1991 0.0377 0.0239 0.0002 0.3702
74   773 0.0423 0.0230 0.0008 0.5324
75   802 0.0343 0.0204 0.0006 0.5281
```

```
76   616 0.0346 0.0221 0.0013 0.2329
77  1248 0.0282 0.0193 0.0012 0.2621
78  2690 0.0241 0.0164 0.0008 0.5655
79  2355 0.0170 0.0121 0.0004 0.2941
80  1541 0.0127 0.0095 0.0009 0.1442
81  1336 0.0164 0.0133 0.0009 0.1858
82  1920 0.0222 0.0174 0.0006 0.1835
83  1696 0.0251 0.0187 0.0002 0.2588
```

In [320]:
```
sns.histplot(z_to_err[7], binrange=[0, 0.1])
```

Out[320]: <AxesSubplot:ylabel='Count'>



In [324]:
```
sum([len(z_to_err[k]) for k in z_to_err])
```

Out[324]: 161546

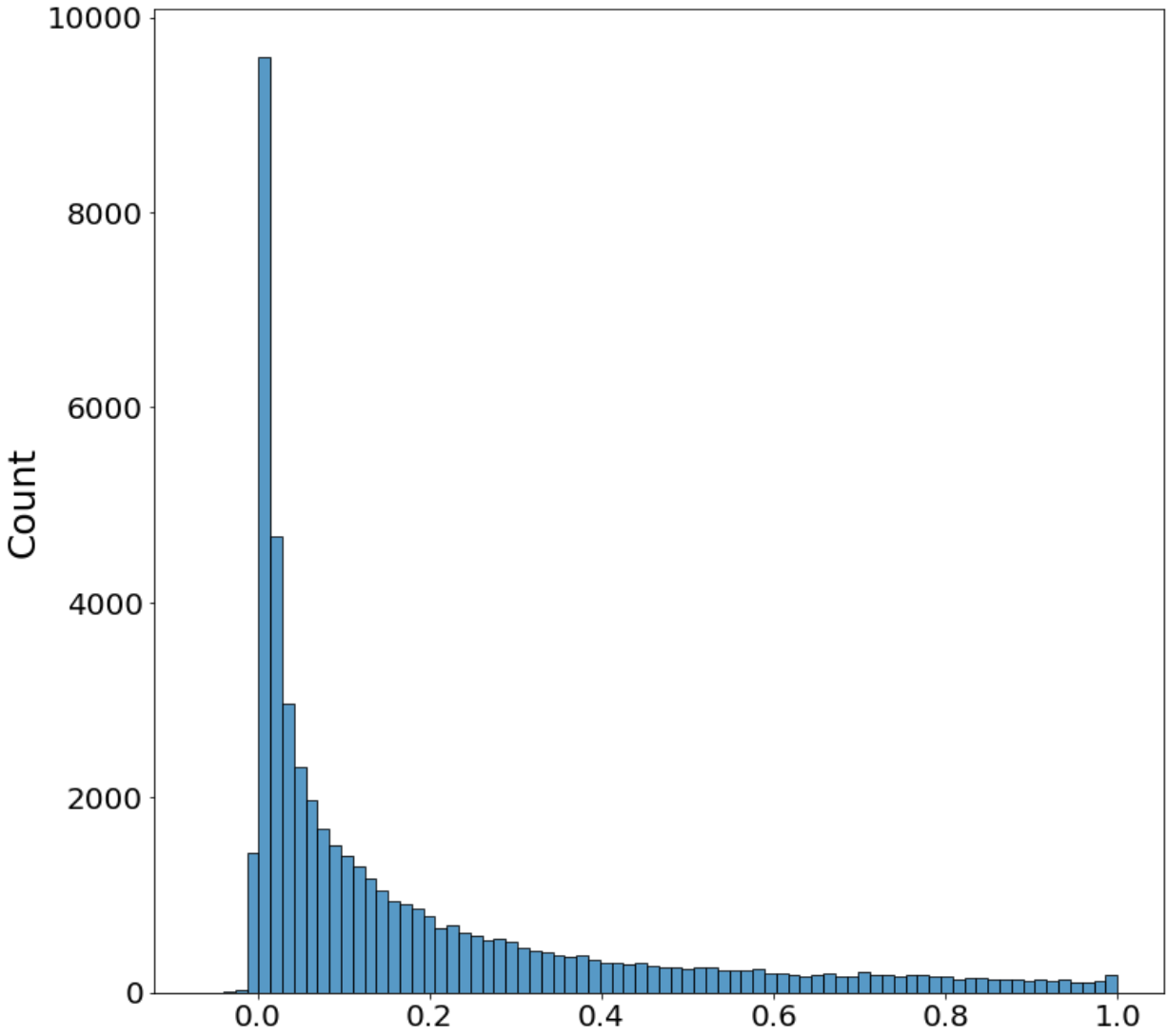
Force cosine distributions with higher epsilon

In [59]:
```
def force_cosine(preds, annos, eps=1e-8, n=None, avg=False):
    cos, natoms = [], []
    n_structures = len(preds["forces"])
    if n is not None:
        n_structures = n
    for i in tqdm(range(n_structures)):
        assert preds["forces"][i].shape[-1] == 3
        assert preds["forces"][i].shape == annos["forces"][i].shape
        # computes cosine similarity b/w predicted and true forces
        # we sum it up for all atoms per structure and keep track
        # of the no. of atoms in each structure.
        cos.append(
            torch.cosine_similarity(
                preds["forces"][i],
                annos["forces"][i],
                eps=eps).sum().item())
        natoms.append(preds["forces"][i].shape[0])

    # if `avg` is True, return a single stat averaged over all atoms
    # from all structures.
    if avg == True:
        return np.sum(cos) / np.sum(natoms)
    # if `avg` is False, return a single stat for each structure
    # averaged over all atoms.
    else:
        return np.array(cos) / np.array(natoms)

cos = force_cosine(reordered_preds, tensorized_annos, eps=1e-1, avg=False,
```

```
100%|██████████| 50000/50000 [00:01<00:00, 25294.44it/s]
```

In [60]:
```
sns.histplot(cos)

np.mean(cos), np.median(cos)
```

Out[60]: (0.1909686219991328, 0.0865661903327278)



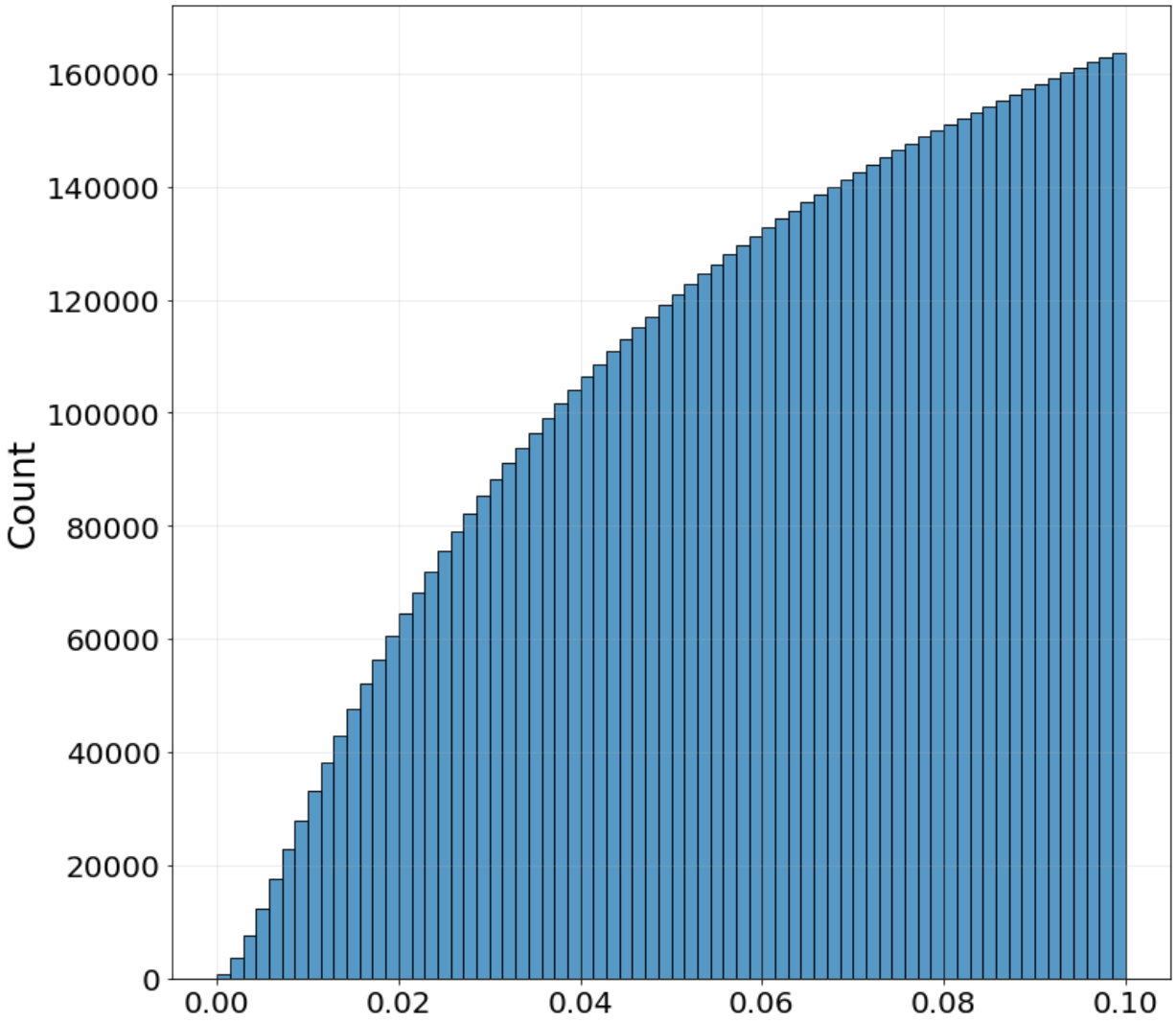
Per-atom force norm distributions

In [96]:
```
def force_norm_per_atom(vecs, n=None, avg=False, p=2):
    norms = []
    n_structures = len(vecs["forces"])
    if n is not None:
        n_structures = n
    for i in tqdm(range(n_structures)):
        assert vecs["forces"][i].shape[-1] == 3
        # compute p-norm of force vectors for each atom
        # in each structure.
        # norms += torch.norm(vecs["forces"][i], p=p, dim=-1)
        norms += torch.norm(vecs["forces"][i], p=p, dim=-1).tolist()
    if avg == True:
        return np.mean(norms)
    else:
        return norms

gt_norms = force_norm_per_atom(tensorized_annos, n=10000, avg=False)
```

```
100%|██████████| 10000/10000 [00:00<00:00, 79789.37it/s]
```

In [100]:
```
plt.grid(alpha=0.25)
sns.histplot(
    gt_norms,
    binrange=[0, 0.1],
    cumulative=True,
    stat="count",
)


np.mean(gt_norms), np.median(gt_norms), len(gt_norms), len(np.array(gt_norm
```

Out[100]: (0.1660604239088742, 0.05329478532075882, 246729, 0.6637039018518294)



NeurIPS ablations plots

In [401]:
```
data = json.load(open("neurips21_spinconv_ablations_data/ablations_forces_m
print(data.keys())
```

```
dict_keys(['data'])
```

In [402]:
```
for d in data["data"]["project"]["runs"]["delta"]:
    print(d["run"]["displayName"])
```

```
oc20-energy-grid12-8-32-s1-67
oc20-force-noconv-48-64
oc20-force-sphhar-48-62
oc20-energy-small-32-66
oc20-force-grid-48-59
oc20-force-small-48-63
oc20-energy-sphhar-32-65
```

In [403]:
```
data["data"]["project"]["runs"]["delta"][0]["run"].keys()
```

Out[403]:
```
dict_keys(['id', 'name', 'displayName', 'updatedAt', 'readOnly', 'framewo
rk', 'notes', 'github', 'group', 'jobType', 'createdAt', 'heartbeatAt',
'commit', 'host', 'state', 'shouldStop', 'groupCounts', 'stopped', 'defau
ltColorIndex', 'sweep', 'agent', 'user', 'tags', 'benchmarkRun', 'runInf
o', '__typename', 'config', 'summaryMetrics', 'systemMetrics', 'sampledHi
story'])
```

In [404]:
```
data["data"]["project"]["runs"]["delta"][0]["run"]["sampledHistory"]
```

Out[404]:
```
[[{'_step': 5001,
   'train/epoch': 0.0011948535999713292,
   'val/forces_mae': 0.07671594776795981},
  {'_step': 10000,
   'train/epoch': 0.002389229354071844,
   'val/forces_mae': 0.07370018722800437},
  {'_step': 15001,
   'train/epoch': 0.003584082954043173,
   'val/forces_mae': 0.07053585164322876},
  {'_step': 20000,
   'train/epoch': 0.004778458708143688,
   'val/forces_mae': 0.06883869549320838},
  {'_step': 25000,
   'train/epoch': 0.00597307338517961,
   'val/forces_mae': 0.06766313022402538},
  {'_step': 30001,
   'train/epoch': 0.00716792698515094,
   'val/forces_mae': 0.06532259980091762},
  {'_step': 35000,
```

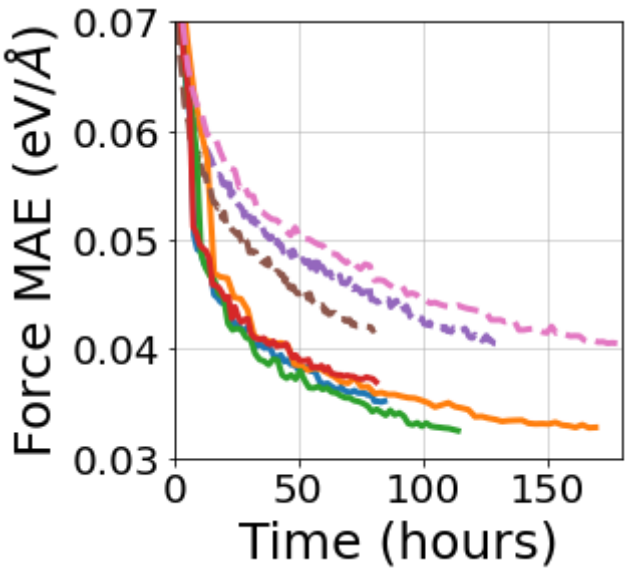
In [346]:
```
# we don't have timestamps recorded.
# [x] manually enter time corresponding to 0.20 epochs for all ablations.
# [x] pull out energy_mae till 0.20 epochs for all ablations.
# [x] set x-timestamps for all energy maes accordingly.
```

In [361]:
```
from collections import OrderedDict

# these are number of hours to 0.20 epochs.
time_to_ep2em1 = OrderedDict([
    ("oc20-force-noconv-48-64", 84.621),
    ("oc20-force-sphhar-48-62", 169.482),
    ("oc20-force-grid-48-59", 114.105),
    ("oc20-force-small-48-63", 81.215),
    ("oc20-energy-grid12-8-32-s1-67", 131.093),
    ("oc20-energy-small-32-66", 80.772),
    ("oc20-energy-sphhar-32-65", 179.924),
])

names = {
    "oc20-energy-small-32-66": "Energy-centric: grid (12x8), small",
    "oc20-energy-sphhar-32-65": "Energy-centric: w. spherical harmonics",
    "oc20-energy-grid12-8-32-s1-67": "Energy-centric: grid (12x8)",
    "oc20-force-small-48-63": "Force-centric: grid (12x8), small",
    "oc20-force-noconv-48-64": "Force-centric: grid no conv",
    "oc20-force-sphhar-48-62": "Force-centric: w. spherical harmonics",
    "oc20-force-grid-48-59": "Force-centric: grid (16x12)",
}
```

In [405]:
```
epoch_limit = 0.2

pdata = {}
for d in data["data"]["project"]["runs"]["delta"]:
    display_name = d["run"]["displayName"]
    sampled_history = d["run"]["sampledHistory"][0]
    pdata[display_name] = [k for k in sampled_history if k["train/epoch"] <
    print("No. of steps for %s: %d" % (display_name, len(pdata[display_name
    log_step_time = time_to_ep2em1[display_name] / len(pdata[display_name])
    for i in range(len(pdata[display_name])):
        pdata[display_name][i]["timestamp"] = (i+1) * log_step_time
```

```
No. of steps for oc20-energy-grid12-8-32-s1-67: 167
No. of steps for oc20-force-noconv-48-64: 63
No. of steps for oc20-force-sphhar-48-62: 63
No. of steps for oc20-energy-small-32-66: 96
No. of steps for oc20-force-grid-48-59: 63
No. of steps for oc20-force-small-48-63: 63
No. of steps for oc20-energy-sphhar-32-65: 96
```

In [409]:
```
metric = "val/forces_mae"
# dns_to_plot = ["oc20-force-noconv-48-64", "oc20-force-sphhar-48-62", "oc2
dns_to_plot = pdata.keys()

fig = plt.figure(figsize=(4, 4))

ax = plt.subplot(111)

for d in dns_to_plot:
    xs = [k["timestamp"] for k in pdata[d]]
    ys = [k[metric] for k in pdata[d]]
    color_index = list(time_to_ep2em1).index(d)
    color_key = list(matplotlib.colors.TABLEAU_COLORS)[color_index]
    if "energy" in d:
        ax.plot(xs, ys, label=names[d], lw=3.0, linestyle="--", color=matpl
    else:
        ax.plot(xs, ys, label=names[d], lw=3.0, color=matplotlib.colors.TAB

ax.set_ylim(0.03, 0.07)
ax.set_xlim(0.0, 180.0)
ax.set_xticks(np.arange(0, 180.0, step=50.))

ax.set_xlabel("Time (hours)")
ax.set_ylabel("Force MAE (eV/$\AA$)")

ax.grid(alpha=0.6)

# plt.legend(loc="right")
# handles, labels = plt.gca().get_legend_handles_labels()
# order = [3, 6, 0, 5, 1, 2, 4]
# lgd = ax.legend([handles[idx] for idx in order],[labels[idx] for idx in o
#                 loc='center right', bbox_to_anchor=(1.6, 1.6))

plt.savefig(os.path.join("figures/ablations_force_mae_v4.png"),
            dpi=150,
            bbox_inches="tight",)
#             bbox_extra_artists=(lgd,),)
```



In [219]:
```
matplotlib.colors.TABLEAU_COLORS
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-219-ba1aeb13a7a3> in <module>
----> 1 matplotlib.colors.TABLEAU_COLORS[0]

KeyError: 0
```