Single Particle IsoNet Tutorial

Single Particle IsoNet (spIsoNet) is designed to correct the preffered orientation effect by self-supervised learning. It iteratively reconstruct missing information in in the missing regions in fourier space. The software requires half maps as input.

Table of contents

1. Installation

We suggest using anaconda environment to manage the IsoNet package.

1. install cudatoolkit and cudnn on your computer.
2. Install pytorch from https://pytorch.org/
3. install dependencies using pip install the dependencies include tqdm, matplotlib, scipy, numpy, scikit-image, mrcfile, fire
4. For example add following lines in your ~/.bashrc

   export PATH=PATH_TO_ISONET_FOLDER/bin:$PATH

   export PYTHONPATH=PATH_TO_PARENT_FOLDER_OF_ISONET_FOLDER:$PYTHONPATH or you can run source source-env.sh in your terminal, which will export required variables into your environment.
5. Now IsoNet is avaliable to use.

The environment we verified are:

1. cuda11.8 cudnn8.5 pytorch2.0.1, pytorch installed with pip.
2. cuda11.3 cudnn8.2 pytorch1.13.1, pytorch installed with conda.

2. Quick run

The default parameter in IsoNet should be suitable for most cases, you can also train with a larger epochs or more iterations to get better result. You can use

```
isonet.py map_refine --help
```

The command to using single particle IsoNet should be

```
isonet.py map_refine halfmap1 halfmap2 maskfile --gpuID 0,1,2,3
```

Here is expected command line output

```
14:32:01, WARNING  [utils.py:7] The isonet_maps folder already exists
 The old isonet_maps folder will be renamed (to isonet_maps~)
14:32:06, INFO     [isonet.py:478] Global resolution at FSC=0.143 is
4.191999816894532
14:32:06, INFO     [isonet.py:482] Limit resolution to 4.191999816894532
for IsoNet missing information recovery. You can also tune this paramerter
with --limit_res .
14:32:06, INFO     [isonet.py:488] calculating fast 3DFSC, this will take
few minutes
14:32:13, INFO     [isonet.py:495] voxel_size 1.309999942779541
14:32:13, INFO     [isonet.py:502] noise_scale: 0.6566901692575242
14:32:13, INFO     [isonet.py:506] processing half map1
14:32:16, INFO     [network.py:228] Port number: 37747
14:32:16, INFO     [map_refine.py:233] Start Iteration1!
14:32:16, INFO     [map_refine.py:235] Start preparing subvolumes!
14:33:48, INFO     [map_refine.py:238] Done preparing subvolumes!
14:33:48, INFO     [map_refine.py:240] Start training!
learning rate 0.0004
True
100%|
Epoch [1/5], Train Loss: 0.2940, Validataion Loss: 0.2801
100%|
Epoch [2/5], Train Loss: 0.2719, Validataion Loss: 0.2724
100%|
Epoch [3/5], Train Loss: 0.2669, Validataion Loss: 0.2683
100%|
Epoch [4/5], Train Loss: 0.2636, Validataion Loss: 0.2666
100%|
Epoch [5/5], Train Loss: 0.2612, Validataion Loss: 0.2649
```

2. individual parameters

###TODO

```
    """
    \ntrain neural network to correct preffered orientation\n
    isonet.py map_refine half1.mrc half2.mrc mask.mrc [--gpuID] [--ncpus]
[--output_dir] [--fsc_file]...
    :param i: Input name of half1
    :param i2: Input name of half2
    :param mask_file: Filename of a user-provided mask
    :param gpuID: The ID of gpu to be used during the training.
    :param ncpus: Number of cpu.
    :param output_dir: The name of directory to save output maps
    :param limit_res: The resolution limit for recovery, default is the
resolution of the map.
    :param fsc_file: 3DFSC file if not set, isonet will generate one.
    :param cone_sampling_angle: Angle for 3D fsc sampling for IsoNet
generated 3DFSC. IsoNet default is 10 degrees, the default for official
3DFSC is 20 degrees.
    :param iterations: Number of iterations.
    :param epochs: Number of epochs for each iteration. This value can be
increase (maybe to 10) to get (maybe) better result.
    :param threshold: Threshold to make 3DFSC volume binary. We usually do
not use it.
    :param n_subvolume: Number of subvolumes
    :param crop_size: The size of subvolumes, should be larger then the
cube_size
    :param cube_size: Size of cubes for training, should be divisible by
16, eg. 32, 64.
    :param mixed_precision: This option will greatly speed up the training
and reduce VRAM consumption, often doubling the speed for GPU with tensor
cores. If you find that this option does not improve speed, there might be
a mismatch for cuda/cudnn/pytorch versions.
    :param batch_size: Size of the minibatch. If None, batch_size will be
the max(2 * number_of_gpu,4). batch_size should be divisible by the number
of gpu.
    :param acc_batches: If this value is set to 2 (or more), accumulate
gradiant will be used to save memory consumption.
    :param learning_rate: learning rate. Default learning rate is xx while
previous IsoNet tomography used 3e-4 as learning rate
    """
```

3 example dataset