A step by step examples

This example processes tomogram containing HIV virus particles through **IsoNet**.

Prepare tomograms and STAR file

First, create a folder for your project. In this folder, create subfolder (in this case subfolder name is tomoset) and move all tomogram (with suffix .mrc or .rec) to the subfolder.

```
mkdir tomoset
mv TS*.rec tomoset/
```

Then run the following command in your project folder to generate a tomogram.star file (in this example, the name of tomogram.star file is hiv_tomo.star)

```
isonet.py prepare_star tomoset --output_star hiv_tomo.star --pixel_size
10.8
```

If you are not using GUI, please use your favorite text editor, such as vi or gedit, to open the **hiv_tomo.star**, and enter one defocus value for each tomogram in the fourth column. This value should be the approximate defocus value calculated for the 0-degree tilt images in angstrom. After editing, your star file should look as follows. Note, this value is only for CTF deconvolution, if you want to skip CTF deconvolution step, leave this column as default 0.

```
data_

loop_
_rlnIndex #1
_rlnMicrographName #2
_rlnPixelSize #3
_rlnDefocus #4
_rlnNumberSubtomo #5
1       tomoset/TS01-wbp.rec    10.800000       38838.257812    100
2       tomoset/TS43-wbp.rec    10.800000       25292.275391    100
3       tomoset/TS45-wbp.rec    10.800000       30169.785156    100
```

CTF Deconvolve

This step not only reduces the CTF artifact but also enhances the low resolution signal making training easier. This step is optional and can not be performed for tomograms acquired with phase plate.

Type the following command in the terminal in your project folder. It will generate deconvolved tomograms in **hiv_deconv** folder.

```
isonet.py deconv hiv_tomo.star --snrfalloff 0.7 --deconv_folder hiv_deconv
```

If the command runs successfully, you will get the following terminal output:

```
######Isonet starts ctf deconvolve######

tomoset/TS01-wbp.rec angpix: 10.8 defocus 3.8838257812 snrfalloff 0.7
deconvstrength 1.0
deconvolved map is saved as  hiv_deconv/TS01-wbp.rec
time consumed:  14.191490888595581

tomoset/TS43-wbp.rec angpix: 10.8 defocus 2.5292275391 snrfalloff 0.7
deconvstrength 1.0
deconvolved map is saved as  hiv_deconv/TS01-wbp.rec
time consumed:  8.547893047332764

tomoset/TS45-wbp.rec angpix: 10.8 defocus 3.0169785156 snrfalloff 0.7
deconvstrength 1.0
deconvolved map is saved as  hiv_deconv/TS01-wbp.rec
time consumed:  8.76533579826355

######Isonet done ctf deconvolve######
```

Generate Mask

To exclude the areas that are devoid of sample, we apply a binary sampling mask to each tomogram. This step is optional but will improve the efficiency of the network training.

By running the following command, 3D mask volumes for each tomogram will be generated and stored in the **hiv_mask** folder. Default parameters will give you a good enough mask.

```
isonet.py make_mask hiv_tomo.star --mask_folder hiv_mask --
density_percentage 50 --std_percentage 50
```

If this command works properly, you will find the mask file, when opened with your favorite mrc image viewer such as 3dmod, covers the areas of your sample of interest. Both this step and CTF deconvolve step will modify your tomogram star file (**hiv_tomo.star**):

```
data_

loop_
_rlnIndex #1
_rlnMicrographName #2
_rlnPixelSize #3
_rlnDefocus #4
_rlnNumberSubtomo #5
```

```
_rlnSnrFalloff #6
_rlnDeconvStrength #7
_rlnDeconvTomoName #8
_rlnMaskDensityPercentage #9
_rlnMaskStdPercentage #10
_rlnMaskName #11
1       tomoset/TS01-wbp.rec    10.800000       38838.257812    100
0.700000        1.000000        hiv_deconv/TS01-wbp.rec         50.000000
50.000000       hiv_mask/TS01-wbp_mask.mrc
2       tomoset/TS43-wbp.rec    10.800000       25292.275391    100
0.700000        1.000000        hiv_deconv/TS43-wbp.rec         50.000000
50.000000       hiv_mask/TS43-wbp_mask.mrc
3       tomoset/TS45-wbp.rec    10.800000       30169.785156    100
0.700000        1.000000        hiv_deconv/TS45-wbp.rec         50.000000
50.000000       hiv_mask/TS45-wbp_mask.mrc
```

Extract Subtomograms

This step extracts small 3D volumes (here we also call subtomograms) randomly from previously described tomograms or deconvoluted tomograms. If you provide a mask in your tomogram star file, the center of the subtomograms is inside the mask areas. The number of subtomograms to be extracted in each tomogram is defined in the **_rlnNumberSubtomo** column in your tomogram star file. You can edit those as your desired value. Usually, total of 300 subtomograms are sufficient for network training.

The following command takes your tomogram star file as input and Generates subtomograms in a folder named subtomo as well as a file named subtomo.star

```
isonet.py extract hiv_tomo.star
```

The subtomo.star contains information for your subtomograms. **_rlnCropSize** is the size of subtomograms, and **_rlnCubeSize** is the size actually used for network training, You can specify these values in the *extract* command.

```
data_

loop_
_rlnSubtomoIndex #1
_rlnImageName #2
_rlnCubeSize #3
_rlnCropSize #4
_rlnPixelSize #5
1       subtomo/TS01-wbp_000000.mrc     64      96      10.800000
2       subtomo/TS01-wbp_000001.mrc     64      96      10.800000
3       subtomo/TS01-wbp_000002.mrc     64      96      10.800000
4       subtomo/TS01-wbp_000003.mrc     64      96      10.800000
5       subtomo/TS01-wbp_000004.mrc     64      96      10.800000
6       subtomo/TS01-wbp_000005.mrc     64      96      10.800000
7       subtomo/TS01-wbp_000006.mrc     64      96      10.800000
```

```
8       subtomo/TS01-wbp_000007.mrc     64      96      10.800000
9       subtomo/TS01-wbp_000008.mrc     64      96      10.800000
10      subtomo/TS01-wbp_000009.mrc     64      96      10.800000
11      subtomo/TS01-wbp_000010.mrc     64      96      10.800000
12      subtomo/TS01-wbp_000011.mrc     64      96      10.800000
```

Refine

The extracted sub-tomograms and subtomo star file are used as input in this refine step, which iteratively trains networks that fill the missing wedge information (and reduce noise). The output is defined by **the result_dir** parameter, whose default value is "results". In this folder, you will find all the subtomograms in each iteration as well as the network model files with the extension of h5, if this command runs successfully.

it will take about 10 hours for four Nvidia 1080Ti to finish the refine step with the following command:

```
isonet.py refine subtomo.star --gpuID 0,1,2,3 --iterations 30 --
noise_start_iter 10,15,20,25 --noise_level 0.05,0.1,0.15,0.2
```

Once you execute the refine step, you will get the command line output as follows:

```
######Isonet starts refining######

06-11 22:00:55, INFO     Done preperation for the first iteration!
06-11 22:00:55, INFO     Start Iteration1!
06-11 22:00:59, INFO     Noise Level:0.0
06-11 22:01:36, INFO     Done preparing subtomograms!
06-11 22:01:36, INFO     Start training!
06-11 22:01:38, INFO     Loaded model from disk
06-11 22:01:38, INFO     begin fitting
Epoch 1/10
112/112 [==============================] - 106s 944ms/step - loss: 0.1597 -
mse: 0.0726 - mae: 0.1597 - val_loss: 0.1575 - val_mse: 0.0711 - val_mae:
0.1575
Epoch 2/10
112/112 [==============================] - 101s 897ms/step - loss: 0.1543 -
mse: 0.0591 - mae: 0.1543 - val_loss: 0.1617 - val_mse: 0.0726 - val_mae:
0.1617
Epoch 3/10
112/112 [==============================] - 101s 904ms/step - loss: 0.1489 -
mse: 0.0513 - mae: 0.1489 - val_loss: 0.1535 - val_mse: 0.0616 - val_mae:
0.1535
Epoch 4/10
112/112 [==============================] - 101s 903ms/step - loss: 0.1486 -
mse: 0.0489 - mae: 0.1486 - val_loss: 0.1583 - val_mse: 0.0687 - val_mae:
0.1583
Epoch 5/10
112/112 [==============================] - 101s 905ms/step - loss: 0.1467 -
mse: 0.0458 - mae: 0.1467 - val_loss: 0.1482 - val_mse: 0.0478 - val_mae:
```

```
0.1482
Epoch 6/10
112/112 [==============================] - 102s 906ms/step - loss: 0.1449 -
mse: 0.0442 - mae: 0.1449 - val_loss: 0.1472 - val_mse: 0.0487 - val_mae:
0.1472
Epoch 7/10
112/112 [==============================] - 102s 906ms/step - loss: 0.1430 -
mse: 0.0409 - mae: 0.1430 - val_loss: 0.1410 - val_mse: 0.0411 - val_mae:
0.1410
Epoch 8/10
112/112 [==============================] - 102s 908ms/step - loss: 0.1437 -
mse: 0.0408 - mae: 0.1437 - val_loss: 0.1427 - val_mse: 0.0437 - val_mae:
0.1427
Epoch 9/10
112/112 [==============================] - 102s 909ms/step - loss: 0.1413 -
mse: 0.0393 - mae: 0.1413 - val_loss: 0.1415 - val_mse: 0.0387 - val_mae:
0.1415
Epoch 10/10
112/112 [==============================] - 102s 910ms/step - loss: 0.1406 -
mse: 0.0383 - mae: 0.1406 - val_loss: 0.1430 - val_mse: 0.0399 - val_mae:
0.1430
06-11 22:21:04, INFO     Done training!
06-11 22:21:04, INFO     Start predicting subtomograms!
06-11 22:22:53, INFO     Done predicting subtomograms!
06-11 22:22:53, INFO     Done Iteration1!
06-11 22:22:53, INFO     Start Iteration2!
```

You can also continue from the pretrained model for this dataset provided in the link. The following command will use the pre-trained network model as the model of 1st iteration, then predict subtomos and train networks starting from this model.:

```
isonet.py refine subtomo.star --pretrained_model ./pretrained_model.h5  --
gpuID 0,1,2,3
```

Another option is to continues from previous runs, with **continue_from** command. This option allows reading the parameter from previous iteration in '.json' file and continuing from that. For example:

```
isonet.py refine subtomo.star --continue_from results/refine_iter20.json  -
-gpuID 0,1,2,3
```

Predict

During the refinement step, the network models are saved in **result_dir** folder. You can select one and apply it to your entire tomograms in the tomogram star file. For example:

```
isonet.py predict tomograms.star ./results/model_iter40.h5 --gpuID 0,1,2,3
```

This process may take a few minutes to predict all tomograms in the tutorial dataset. You can also use **tomo_idx** to tell the program which tomogram(s) you want to predict.

Now, We now the missing wedge corrected tomograms in the corrected_tomos folder.