IsoNet Tutorial

USTC
CNSI

publication not available yet

Table of Contents

1. Introduction

IsoNet stands for for ISOtropic reconstructioN of Electron Tomography. It trains deep convolutional neural networks to reconstruct meaningful contents in the missing wedge for electron tomography, and to increase signal-to-noise ratio, using the information learned from the original tomogram. The software requires tomograms as input. Observing at about 30A resolution, the IsoNet generated tomograms are largely isotropic. For installation guide, please find Readme.md on github page: https://github.com/Heng-Z/IsoNet/

2. Quick going through

The following describes the basic steps of running the program and generate missing wedge corrected tomograms using command line from the demo dataset. This dataset contains 8-times binned tomograms reconstructed from three tilt series in EMPIAR-10164. You can download the tomograms from Google drive.

After completing this quick tutorial, you are able to achieve the results shown in the following image. For detailed description of each commands, please refer to **Individual tasks**. Among the following steps, only refine and prefect steps are computational extensive and can use GPU.

Let's get it started.

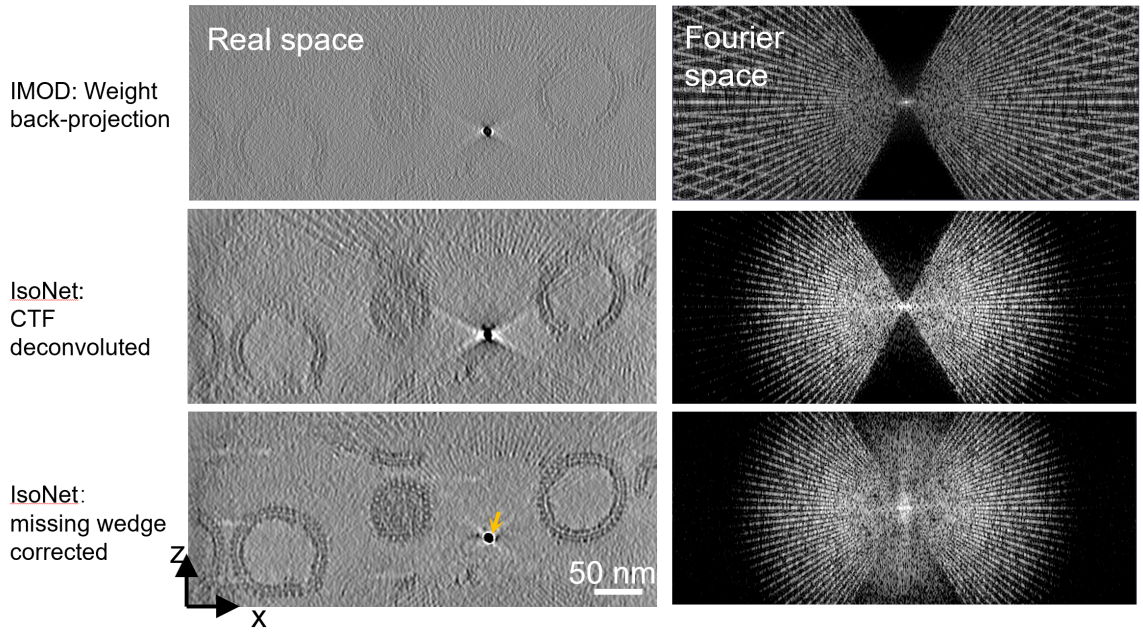

Fig. 1. XZ slice views of HIV tomograms reconstructed with weighted back projection (upper), CTF deconvoluted tomogram (middle) and missing-wedge corrected tomograms (bottom), with their Fourier transforms on the right. Orange arrow indicates a gold bead.

2.1 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 values for each tomogram in fourth column. This value should be approximate defocus value calculated for the 0 degree tilt images in angstrom. After editing, your star file should looks as the 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
```

2.2 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######
```

2.3 Generate Mask

To exclude the areas that 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 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
```

2.4 Extract Subtomograms

This step extracts small 3D volumes (here we also call subtomograms) randomly from previous described tomograms or deconvoluded tomograms. If you provide a mask in your tomogram star file, the center of the subtomograms are 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 300 subtomograms are sufficient for network training.

The following command takes your tomogram star file as input and generate 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 size of subtomograms, and **_rlnCubeSize** is the size actually used for network training, You can specify these values in 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
```

2.5 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 in **result_dir** parameter, whose default value is "results". In this folder, you will find all the subtomograms in each iterations as well as the network model files with 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 pretrained 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 continue from that. For example:

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

2.6 Predict

During the refine 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.

3. Individual tasks

3.1 Prepare tomograms and STAR file

IsoNet uses star file format to store information of tomograms. This file can be prepared using **isonet.py prepare_star** command prior to the subsequent processing. To do so, user should prepare a folder containing all tomograms. Binning the tomograms to pixel size of about 10 A is recommended since the target Z axis resolution should be about 30A. Too large (>20A) or too small (<5A) pixels might reduce the efficiency of IsoNet network training. The default **pixel_size** parameter is 10A. We typically use a folder containing 1 to 5 tomograms as input.

The input tomograms can be either reconstructed by SIRT or WBP algorithm. The tilt axis is y axis and recommended tilt range is -60 to 60 degrees, without x axis tilt, while other tilt ranges might also work. The tilt series can be collected with any tilt schemes, continues, bidirectional or dose-symmetric.

If you do not want to perform CTF deconvolution, especially when the tomogram is acquired by phase plate or the tomogram is CTF corrected, the **defocus** can be left as 0. Otherwise, you can use the **defocus** parameter to set one defocus value for the tomograms. This value should be the defocus value of zero tilt image. We do not consider defocus variation across different tilted images in this version of IsoNet, since it does not target for high resolution currently. When you have multiple tomograms in the folder, the **defocus** parameter (in angstrom) for each tomogram should be adjusted in the star file with your text editor, after the tomogram star file has been generated with **isonet.py prepare_star** command.

3.2 CTF deconvolve

Given the defocus values in the tomogram star file, CTF deconvolution can be performed by applying a Weiner filter to each tomogram. This step not only reduces the CTF artifact but also enhances the low resolution signal so that it is more easy to train the network. This step can be skipped for tomograms acquired with phase plate. This step is similar to the CTF deconvolve in Warp software.

3.2.1 Deconvolution parameters

Two parameters, **snrfalloff** and **deconvstrength**, are worthy to tuned in this step to so that the output tomograms have visually highest contrast. If these parameters are not set in the command, the values in the star file will be used; If the star file does not contain these parameters, default 1.0 will be used for both parameters. The effect of these two parameters are shown in the following figures. You can also specify **tomo_index**, e.g. 1,3-4 , so that only the specific tomogram or tomograms will be processed. Another parameter **hipassnyquest** applies a high pass filter at very low frequency, changing this parameter might be helpful if the tomograms are too blurry.

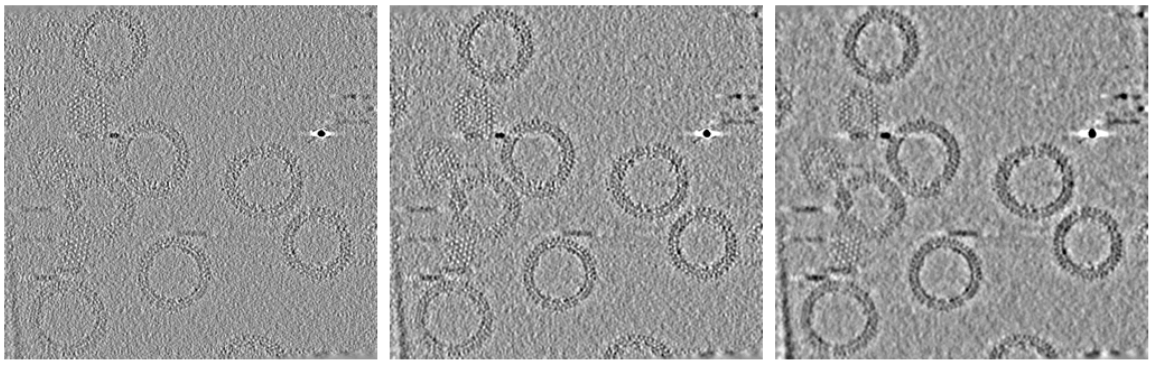

Fig. 2. 2D slices of CTF deconvolved tomograms with different snrfalloff parameters. Left: snrfalloff=0.5; middle: snrfalloff=1; right:snrfalloff=1.5

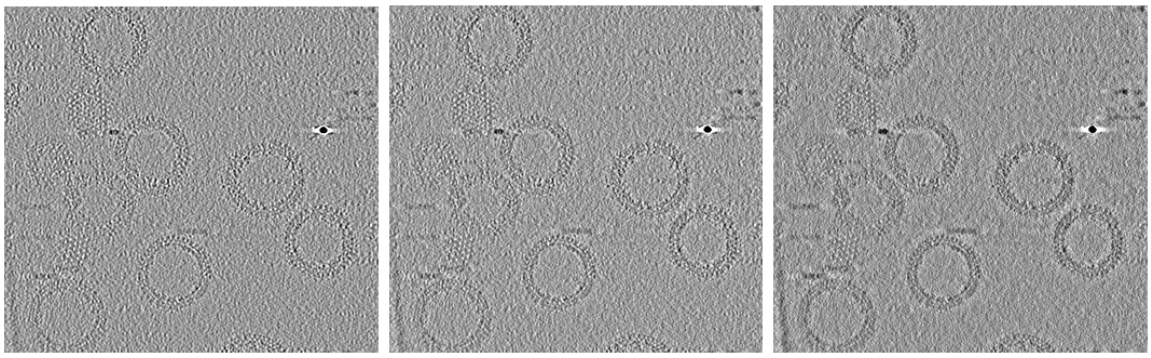

Fig. 3. 2D slices of CTF deconvolved tomograms with different deconvstrength parameters. Left: deconvstrength=0.5; middle: deconvstrength=1; right:deconvstrength=1.5

3.2.2 Parallel processing

Multiple CPUs can be used for CTF deconvolution, number of CPU is set as parameter **ncpu**, the default value for **ncpu** is four. The program crops the tomogram in to multiple tiles for multiprocessing and assembly them into one. The number of tiles in z, y, x directions are defined in parameter **tile**, the default value for tile is (1,4,4). This option may induce artifacts along edges of tiles, when that happen, you may disable tile-based multiprocessing or use larger **overlap_rate**.

3.3 Generate mask

To obtain a training dataset, sub-tomograms are randomly extracted from tomograms. However, when the sample in a given tomogram is sparsely distributed, most of the extracted sub-tomograms will not contain meaningful information. Therefore, the performance of network training might be reduced, although in most cases, you can still get reasonable result from training without mask.

To help with getting more meaningful training dataset, we introduce the **make_mask** module, which created three types of mask: 1 Pixel intensity mask, 2. Standard deviation mask, 3. mask that crop out top and bottom parts of tomograms

The final mask created by the command is the **intersection** of these three types of the mask.

3.3.1 Pixel intensity mask

This type of mask will mask out empty areas based on their relatively low local pixel density. A maximum mask will first suppress the noise with a 3D Gaussian filter and then apply a 3D sliding window maximum-density filter to the tomograms. The window size of the maximum filter is defined by **patch_size** parameter. This size can be increased if the tomograms are too noisy.

In this filter tomogram, the areas with relatively smaller density values will be deemed as empty space and can be excluded with parameter **density_percentage**, ranging from 0 to 100, meaning the only include this percentage of pixels in the mask.

Usually, lower **density_percentage** value should be used when having tomograms with sparsely distributed samples. However, this type of mask does not work well when the tomograms does not have uniform background, e.g. darker on one side of the tomogram.

When you don't want to use this mask, set the **density_percentage** value to 100.

3.3.2 Standard deviation mask

Recognizing that the "empty" regions of the tomograms are often areas with low standard deviation (STD), this standard deviation mask is designed to exclude the low STD areas.

To do so, we calculated the STD of a volume centered at the evaluating pixel (local STD). The size of the volume to measure local STD is defined by the parameter **patch_size**, this parameter is also used to define size of max filter in pixel intensity mask.

All the pixels with STD ratio larger than the **std_percentage**% of all pixels will be included in the mask. When you don't want to use this mask, set the **std_percentage** value to 100.

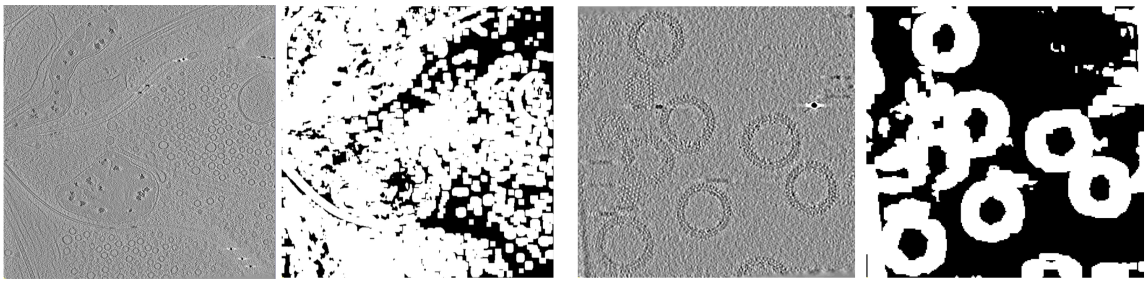

Fig. 4. XY slices of tomograms and corresponding masks. Both using density_percentile=50 and std_percentile=50

3.3.3 Crop out top and bottom parts of tomograms

Sometimes the top and bottom parts of tomograms are devoid of sample. **z_crop** parameter is designed to exclude the top and bottom regions of tomograms along z axis. For example, **--z_crop 0.2** will mask out both the top 20% and bottom 20% region along z axis.

3.4 Extract

This step randomly extract the subtomograms from the tomograms in the input tomogram star file. The output of this command is a folder containing all the subtomograms and a star file containing the information of those subtomograms.

The number of subtomograms to be extracted for each tomogram should be written in the tomogram star file in the column of (_rlnNumberSubtomo). User can modify this number in the star file. Ideally, the smaller the masked area, the smaller this number should be. Usually total 300 subtomograms are sufficient for refine.

If mask files are provided in the input tomogram star file, the centers of the subtomograms are always inside the mask regions. If CTF deconvoluted tomograms are provided in the tomogram file, those tomograms will be used for the subtomogram extraction, unless the **use_deconv_tomo** parameter is set to False.

The parameter **cube_size** is the size of the cubic volume for training, not the size of extracted subtomograms. The actual size of the extracted subtomograms is 1.5 times of **cube_size**. The **cube_size** should be divisible by 8 and is usually limited by the GPU memory. 64 or 96 is often good estimation for **cube_size**. If you encountered out-of-memory (OOM) problem during network training, reducing this value is one of the choices.

3.5 Refine

This process iteratively trains neural networks to fill the missing wedge information using the same tomograms whose missing wedge artifacts were added to other directions. The denoising module can also be enabled in this step, making the network capable of both reducing noise and recover missing wedge. After refine, the subtomograms neural network model in each iteration are saved in the results folder. The network models with suffix of ".h5" can be used for the prediction step.

3.5.1 Optimizing computation resources

At the beginning of each iteration, IsoNet will process the subtomograms, such as rotating, cropping and applying missing wedge. Only CPUs are used for those processes. The parameter **preprocessing_ncpus** is used to define how many CPU cores are used for this preparation step.

User have to specify **gpuID**, e.g. 0,1,2,3 , for the network training, so that 3D volumes will be distributed across those GPUs. Information of available GPUs can be found through commond: *nvidia-smi*. In general, Using more GPUs reduces GPU memory requirement for each GPU and current network training can not be performed across multiple computer machines.

The **data_folder** is the folder that temporally stores the training and test data pairs. The

files in that folder will be updated every iteration. Setting it to a faster drive such as SSD or memory file system will presumably increase the speed of network training, though not fully bench-marked on the developer's side.

3.5.2 Optimizing training speed

One **iteration** of refine contains three steps: training data preparation, network training and subtomogram prediction. The model will be refined in iteratively based on previous prediction results. In practice, total 10 to 20 iterations are typically used for refine without denoising. Please refer to denoising section for more details.

The training step is divided into several **epoch**s. Each epoch will traverse through the randomly shuffled data set. The default value 10 for the number of epochs is usually sufficient. Training data pairs are grouped into batches to feed into each epoch. The **batch_size** should be divisible to the number of GPU, so that the data can be distributed into multiple GPUs. If you are using multiple GPUs and **batch_size** is not set, the default value is two times of the number of GPU. If you are using single GPU, the default **batch_size** is four. We tested **batch_size** of 4-12 can result in good performance, too large **batch_size** might lead to OOM error. **step_per_epoch** defines how many batches to be processed in one **epoch**. A value between 100 to 300 are recommanded for **step_per_epoch**. If this value is not set by user, the default **step_per_epoch** is min(number_of_subtomograms * 6 / batch_size , 200)

3.5.3 Denoising

The **noise_start_iter** parameter defines when the denoising will be apply to the training. Usually, it is the iteration in which undenoised training the expected to be converged. The **noise_level** is the ratio of standard deviation of the additive noise compare to the original data. Once the denoise was applied, the mean absolute error loss of the training will be increased. If the **noise_level** is too high, the training might failed, as indicated by the extremely large losses.

You can set both **noise_start_iter** and **noise_level** with multiple values. So that you can gradually increase the **noise_level** during the training. We recommend you using multiple values of the **noise_level** and **noise_start_iter**.

For example, in the following command, the **iteration** parameter is set to 30, **noise_level** is set to 0.1,0.2 , and **noise_start_iter** is 11,21. Then the first 10 iterations are trained without denoising, 11-20 iterations are trained with noise level 0.1, 21-30 iterations are trained with noise level 0.2. You can use the neural network model from iteration 10, 20, 30 to

predict the same tomogram and distinguish which level of denoising is best for your tomograms.

```
isonet.py refine subtomo.star --iter 30 --noise_level 0.1,0.2 --
    noise_start_iter 11,21
```

3.5.4 Network structures

IsoNet allows user to modify the network structures by the input arguments. For example, it might be useful to increase or decrease the size of network by increase or decrease **unet_depth**, though this is not recommended unless user want to venture into performance of different networks. Please note that these values will be ignored when using **pretrained_model**.

If **normalize_percentile** is set True, tomograms will be normalized by percentile, which scale the sub-tomograms in a range approximately from 0 to 1. If this is set False, the subtomograms will be normalized to have a mean of zero and a standard deviation of 1.

3.5.5 Continuing using previous trained network

If you want to continue with model from previous iterations of refine, you can specify the **continue_from** argument, which takes the '.json' file that generated at each iteration.

For example:

```
isonet.py refine subtomo.star --continue_from ./results/
    refine_iter30.json --gpuID 0,1,2,3 --iterations 50
```

If you already have a trained network model (with file extension of .h5), instead of '.json' file , you may want to choose **pretrained_model** option:

For example:

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

This command enables using your pretrained model to predict the subtomograms of the first iteration. Starting with the second iteration, you are refining this model using the subtomograms in the subtomo.star

3.6 Predict

This module applies the trained network model to tomograms to restore the information in the missing wedge region of tomograms. It takes the tomogram star file, which can be generated with **isonet.py prepare_star** command, and a trained network model (with .h5 file extension) as input. The input tomograms in the tomogram star file are typically the exact tomograms used for training or other tomograms with the similar sample and imaging condition. If the network is trained with CTF deconvolved tomograms, the tomograms used for predict should also be CTF deconvolved.

This step is much faster than refine step. **gpuID** defines which GPU(s) will be used for predicting, e.g. 0,1,2,3. If this parameter is not set, one cpu will be used for predict, which could take much longer time than using GPU.

To fit the tomogram into the GPU memory, one tomogram is divided into multiple tiles for the missing wedge correction and using the overlap tile strategy to prevent the artifact during the montaging the tiles. To implement this strategy, the **crop_size** should be larger than the **cube_size**. The **cube_size** and **crop_size** is suggested to be consistent with the training settings. If **crop_size** is not large enough, you may observe artifacts of grids between the adjacent tiles.

The **batch_size** defines number of the cubic tomogram volumes grouped together for network predicting, this value should be divisible by the number of GPU. Larger **batch_size** will save more predicting time but occupy larger your GPU memory. **normalize_percentile** should be the same with that parameter in refine.

```
isonet.py predict tomogram.star path_to_network_model --gpuID
    0,1,2,3 --cube_size 80 --crop_size 128
```

3.7 Prepare star file from particles

This step allows generating a star file from a folder containing only subtomogram files. The subtomogram files should be in mrc format, with extension of ".mrc". The generated star file can be used as input for refine.

This step is not in the main workflow of IsoNet, but might be helpful if you already have subtomogram extracted, possibility through other softwares. For example, if you are only interested in processing subtomograms of a particular protein, you can manually pick and extract them, then prepare star file with this command. We also recommend these subtomograms are downing-scaled to 5-15 A/pixel and CTF deconvoluted if those are not acquired with phase plate.

This command works as follows:

```
isonet.py prepare_subtomo_star folder_name [--output_star] [--
    cube_size]
```

Where "folder_name" is the folder containing subparticles. The default output star file is "subtomo.star".

And the "cube_size" is size of the cubic volumes used for training. This values is smaller than the size of subtomogram and should be divisible by 8, eg. 32, 64. If this value isn't set, "cube_size" is automatically determined as int(subtomo_size / 1.5 + 1)//16 * 16

4. GUI

This section briefly described ISONET graphic user interface (GUI). You can watch our tutorial video for the details on how to use GUI.

The GUI can be started by the following command:

```
isonet.py gui
```

This software mainly have 3 pages, they are **Preparation**, **Refinement** and **Prediction**.

1. Preparation includes the reprocess steps to prepare the dataset to be trained in the later Refinement step.
    - Deconvolve ctf: increase the contrast of tomograms with no VPP applied
    - Generate mask: only mask out the interested region in the tomogram
    - Subtomograms extraction: generate training data set
2. Refinement will train a neural network to learn how to correct the missing wedge effect.
3. Prediction help users to generate corrected tomogram based on the model they trained in the Refinement step.

Isonet take **.star** file as its input file, which follows the convention from *Relion*. The GUI will automatically read the **tomograms.star** as default if it exists in the current folder.

In preparation tab, a input table can help user create or load their own **.star**. One can click insert to add a new row into the table. For tutorial dataset of HIV, you need to specify

MicrographName (reconstructed tomogram), pixel size, estimated defocus value for your 0 degree image. You can select one row by click the index of the row on the left-most region. After selected, you can duplicated it or delete it by clicking insert or Delete button on the right. '3dmod view' helps user to visualize selected tomograms and/or masks.

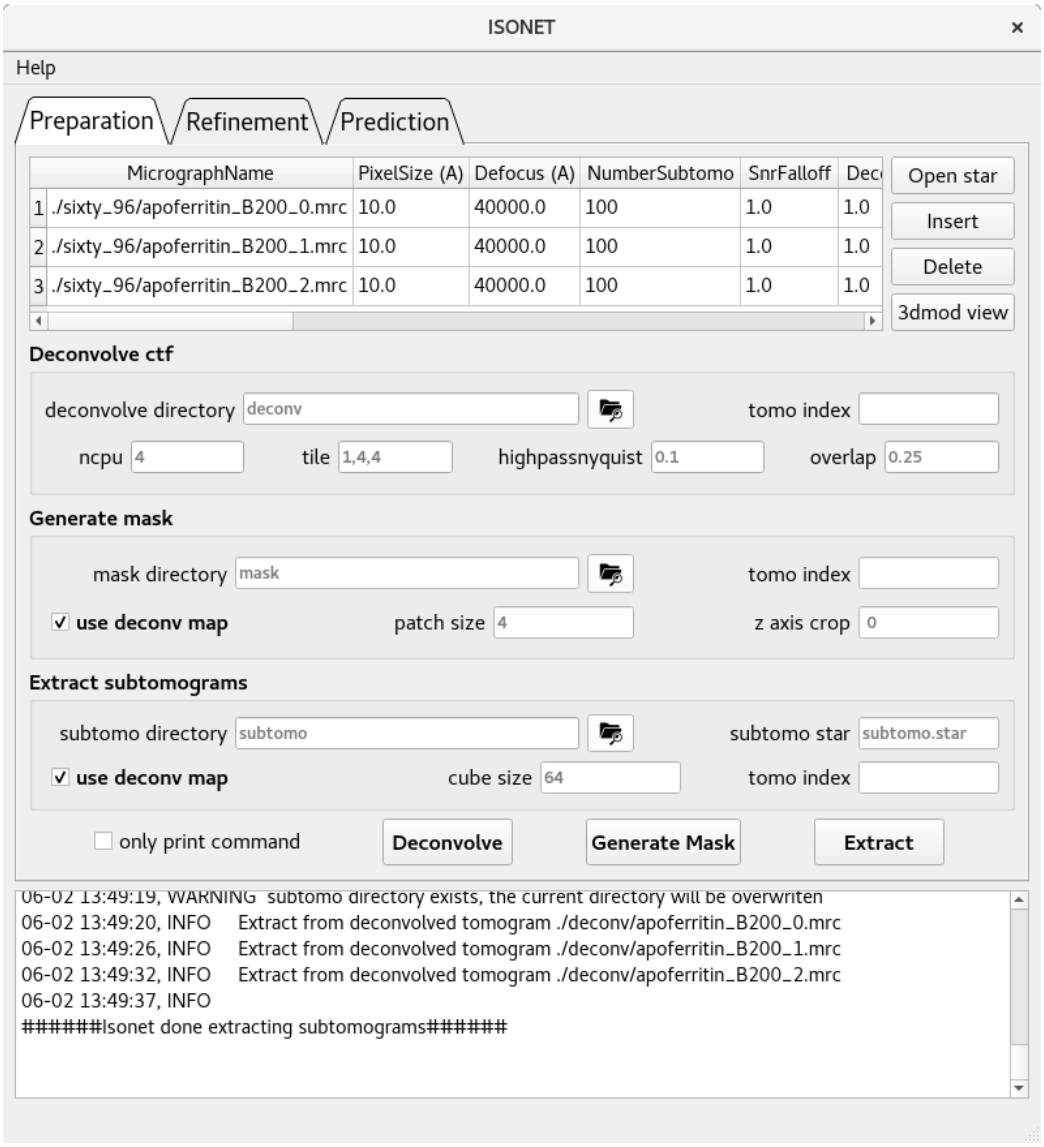

Fig. 5. Preparation page of ISONET GUI.