# Run a single ptychography reconstruction:

* Files needed:
  + Ptychography software package from Yi Jiang’s fold\_slice repo on Github: <https://github.com/yijiang1/fold_slice>.
  + Job submission related files, they can be downloaded from <https://cornell.box.com/s/3vkqwdfboexuzw4fsjnvnwf3b8kq9dy7>. Put the four .m files under fold\_slice/ptycho of the ptychography software.
  + 4D Ptychography in a .mat file, with the 4D array variable name being ‘cbed’, the first two dimensions are k-space.
  + Test data for mixed state ptychography can be found on <https://www.nature.com/articles/s41467-020-16688-6>
  + Test data for multislice ptychography can be found on <https://data.paradim.org/doi/ssmm-2j11/>. The 4D array is named as ‘dp’, rename it as ‘cbed’ and save the data again.

## Information to collect:

* + Absolute path (start with /home/fs01) of the 4D dataset. -> second row of parameter file.
  + Absolute path of the folder to save the results. -> third row of the parameter file.
  + Experiment conditions (convergence angle, scan step size, rotation angle, etc) -> parameter file.
  + Absolute path of the parameter file. -> PARFILE of the .sub file.
  + Absolute path to the fold\_slice/ptycho folder in the ptychography software package. -> EXEPATH of the .sub file.

## For each ptychography reconstruction:

* + Open the parameter file, modify the path on the second and third row to point to the location of the raw data and the directory to save the results. Modify the rest parameter to match experimental conditions.
  + Open the submission file, modify variable EXEPATH to the location of fold\_slice/ptycho, modify variable PARFILE to point to the parameter file. Modify the requested resources accordingly, 1 GPU and 32 CPUs should be good for all the jobs.
  + Connect to ALTAS using ssh, enter the path of the .sub file, submit it with sbatch, wait for the job to finish and collect results from the result saving path.

# Benchmark tests for reconstruction time:

* Multislice ptychography

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Computing resources | RTX4000 + 40 CPUs | A100 + 32 CPUs | ½ A100 + 32 CPUs | 1/3 A100 + 32 CPUs | ¼ A100 + 32 CPUs |
| Time per initialization iteration (sec) | 22.8 | 12.3 | 14.8 | 18.0 |  |
| Time per refine iteration (sec) | 74.7 | 23.7 |  |  |  |

* Processing time per iteration in seconds for different combinations of nlayers and CBED numbers, on single A100 GPU, 128px CBED. If CBED is padded to 256 px, the time consumption will be roughly 4x the value listed here.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 15 layers | 25 layers | 30 layers |
| 64x64 CBEDs |  |  | 17 |
| 128x128 CBEDs | 90 | 170 | 225 |
| 256x256 CBEDs |  |  | 1600 |
| 512x512 CBEDs |  |  | 13500 |

* Mixed state ptychography

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Computing resources | RTX4000 + 40 CPUs | A100 + 4 CPUs | ½ A100 + 4 CPUs | 1/3 A100 + 4 CPUs | ¼ A100 + 4 CPUs |
| Time per iteration (sec) | 0.91 | 0.51 | 0.78 | 1.07 | 1.35 |

* Reconstruction can be more parallel and faster with larger group size, but it also makes the jobs less likely to converge to the correct answer, and more likely to stop early due to fitting errors.

# Run Bayesian optimization to determine the parameters:

## Create a new python virtual environment with dependencies

* + Botorch >= 0.5.1, gpytocrh >= 1.6.0, torch >= 1.9.0+cu111, scipy >= 1.7.3., python >= 3.7
  + Use python3.8 instead of python3 when creating the new virtual environment to install the latest version of botorch.

python3.8 -m venv /home/fs01/cz489/venv/torch

source /home/fs01/cz489/venv/torch/bin/activate

pip install –-upgrade pip

pip3 install torch==1.10.2+cu113 torchvision==0.11.3+cu113 torchaudio==0.10.2+cu113 -f <https://download.pytorch.org/whl/cu113/torch_stable.html>

pip install botorch

## Test pytorch connection to the GPU

There is no GPU on the head node, so we need to submit an interactive job to one of the computing nodes, then run python and import pytorch to see if it is connected to the GPU.

source /home/fs01/cz489/venv/torch/bin/activate

srun -n 1 --gres=gpu:2g.20gb:1 --pty bash

module load cuda/11.5

nvidia-smi # if cuda talks to the GPU, this step should show the GPU status

python

import torch

torch.cuda.get\_device\_name(0) # if python talks to cuda, this step should show the GPU partition name

## Modify the files for Bayesian optimization

* + Download the whole fold\_slice folder with customized files for Bayesian optimization from [https://github.com/CY-Zhang/fold\_slice/tree/main/](https://github.com/CY-Zhang/fold_slice/tree/main/ptycho/ALTAS_submission/BO_test)
  + The files for optimization can be found under fold\_slice/ptycho/ALTAS\_submission/BO\_test. There are three files that needs to be modified for each task: setup.txt, bo\_thread.sub, and ptycho\_thread\_template.sub. The five .py files needs to stay in the same folder as the .txt and .sub files.
  + For my personal preference, I keep the five python scripts, the two .sub files, and the setup files in the same folder, and create a subfolder as the result saving path. It is helpful to also make a copy of the setup.txt file in the result saving path, so that in the future, we can check what parameters were used for that optimization run.
  + bo\_thread.sub
    - This is the entry point to the whole Bayesian optimization, we submit this file with sbatch to start the whole process.
    - Line 2, job name, change to customized job name.
    - Line 9, time limit, the optimization will keep running until reaching the time limit. For multislice ptychography optimization with 5 layers, > 10 hrs are needed. For mixed-state ptychography, 2-4 hours will be sufficient.
    - Line 14, change the path to the virtual environment path with botorch and pytorch installed.
  + ptycho\_thread\_template.sub
    - This is a template file for each ptychography run in the paralleled optimization process. The main process will create separate job submission files for multiple ptychography reconstruction following the template given here.
    - Line 2, job name, change the part before *&THREAD\_NUM&* to a customized name.
    - Line 9, time limit, use the same time limit as bo\_thread.sub.
    - Line 16, change the *EXEPATH* variable to the ptycho folder under fold\_slice in your own path.
    - Line 17, change the *PARFILE* variable to the path where the .sub, .txt, and .py files are kept.
    - Line 24, change *run\_multislice\_new* to *run\_mixed\_state* if running single layer mixed state ptychography.
  + setup.txt
    - Line 2, change the path of raw\_data to the path of the .mat file.
    - Line 3, specify the folder to keep all the results. For each set of parameters being tested, it creates a tif image in the result folder which is the final phase object, and the filename contains the parameter selection and the Fourier error from that set of parameters.
    - Line 4, specify the number of jobs to be run in parallel. The total job number include one job for the BO thread, and njob-1 threads for the ptychography reconstructions. Make sure the cluster has at least one *1g.10gb* partition left for the BO thread, and njob-1 *2g.20gb* partition left for the ptychography threads.
    - Line 5-30 are the ptychography parameter for the reconstruction. For parameter that needs to be optimized, use two numbers for the initial guess and search range. For the fixed parameters, use one number for the value. Currently, the search only work if the parameter is a continuous number, which means it does not work for integer parameters such as number of layers, number of probe modes, or binary parameters such as variable\_probe.
    - If running multislice ptychography, make sure the nlayers and thickness parameters are included. For mixed state ptychography, these two parameters will be ignored even if they are listed in the setup file.

## Submit the Bayesian optimization job.

* + Connect to the cluster in a terminal, go to the folder with all the .sub files, and submit the job with *sbatch bo\_thread.sub*.
  + During the optimization or after the whole optimization is done, you can go to the result folder and check the phase object image with the largest error in the filename (Here BO maximizes the opposite value of log(error), so the one with largest error corresponds to the best reconstruction), the parameters in the filename is the optimized parameters.

# Tips

* Sometimes, the parameters that produces the lowest Fourier error might be using extremely small convergence angle and produce an object that is completely blurred. So, besides a good reconstruction error, also make sure to check if the phase object image shows expected lattice structure.
* When using very dense layers (like 5 A layer thickness), set *variable\_probe* to False could be helpful in preventing the job from failing. It is helpful if the job failed while running with all the previous results look reasonable.
* If a set of parameters cannot give good results/failed when using layer thickness = 10 A, it is not likely to get any better results by using 5 layers at the beginning and gradually include more layers.
* In general, the reconstruction should converge within 500 iterations if we used the correct parameters.