Requirements for running the pyNX based ptychography codes

1. Linux VM with cuda installed
2. Install anaconda
   1. wget <https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh>
   2. bash Anaconda3-2020.02-Linux-x86\_64.sh
3. Install pycuda(python interface to talk to cuda codes)
   1. pip install pycuda
4. Install other dependencies
   1. pip install --upgrade pip
   2. pip install setuptools wheel --upgrade
   3. pip install numpy cython scipy matplotlib ipython notebook scikit-image ipywidgets ipympl
   4. pip install h5py hdf5plugin h5glance silx fabio
   5. pip install scikit-cuda
   6. pip install <http://ftp.esrf.fr/pub/scisoft/PyNX/pynx-latest.tar.bz2>
5. ipythonwidgets optional package for plotting and dynamic interactions
   1. conda install -c conda-forge ipywidgets
6. conda install -n base -c conda-forge widgetsnbextension
7. jupyter labextension install @jupyter-widgets/jupyterlab-manager
8. jupyter nbextension enable --py widgetsnbextension
9. jupyter lab clean
10. jupyter lab build

Now we can run the pyNX codes on VM using the jupyter-lab environments or python scripts.

What we did?

We set up a pre-existing framework created by ESRF for Fourier ptychography on cloud-based architectures which is Azure in our case and provide a proof-of-concept of parallelization of post-processing in X-ray diffraction studies. Pynx is a high-level library with cuda implementations of the algorithms which makes it really fast in comparison to previously existing ptychography codes (like ptypy and ptychoshelves). We prepared examples to showcase the ptychography codes and capabilities of the framework.

Example 1 (Cat vs Dog.):

We provided Cat image as Amplitude of the Object wave function and Dog image as phase of the same and simulated the diffraction data using the pynx.ptycho.simulation module. Using the simulated diffraction data, we were able to successfully recover the object (phase and amplitude) using the pynx.ptycho framework.

Example 2(polka dot pattern):

We provided a much easier pattern image(polka dots spaced at regular intervals) as Amplitude of the Object wave function and the same image as phase to simulate the diffraction data using the pynx.ptycho.simulation module. Using the simulated diffraction data, we were able to successfully recover the object (phase and amplitude) using the pynx.ptycho framework.

Experiment 1(Downsampling diffraction data using average pooling of pixel intensities):

Using the same image from Example-2, we downsample the simulated diffraction data (not the image but the diffraction intensities). The detector had an original size of 256x256 pixel resolution, and each pixel is a square of size 80microns. That means that each diffraction image/array have the respective dimensions. We downsample the detector patterns from 256x256 to 128x128 by average pooling of 4 pixels into 1 pixel in the output. Results indicate that downsampling using average pooling doesn’t facilitate the recovery of object phase and amplitude.

Experiment 2:

Using the same image from Example-2, we downsample the simulated diffraction data (not the image but the diffraction intensities). The detector had an original size of 256x256 pixel resolution, and each pixel is a square of size 80microns. That means that each diffraction image/array have the respective dimensions. We downsample the detector patterns from 256x256 to 128x128 by maximum/minimum pooling of 4 pixels into 1 pixel in the output. Results indicate that downsampling using this method doesn’t facilitate the recovery of object phase and amplitude.