I am trying to pre-process 3D volumes from microscopy data, and I want to implement eigen-decomposition using parallelised CPU (8 workers). I have a MacBook Pro M1 Max 64GB so, there is an opportunity to implement certain computations in the GPU (eigen-decomposition is not supported). The methodology is as follows:

1. Normalise the volume and add epsilon to prevent zeros.
2. We need to scan voxel per voxel in 26 different directions (faces, edges, corners). We do this via a structure tensor (3x3). Please elaborate if the tensor structure implementation can be done in GPU. Use the structure tensor library (<https://github.com/Skielex/structure-tensor> )
3. Use only Apple MPS (Pytorch) equivalent libraries. No CUDA, no CUPY or anything like it.
4. We need to implement eigen-decomposition to get eigenvectors and eigenvalues. Then compute anisotropy: fractional, planar, spherical, linear. This needs to be done in CPU but parallelised. For that you need to use:

*# Normalized planar anisotropy (more sensitive)*

CP\_normalized = (lambda2 - lambda1) / (lambda3 - lambda1)

*# Direct eigenvalue ratios*

lambda2\_ratio = lambda2 / lambda3 *# Shows planar structure directly*

Also, make sure that spherical anisotropy does not count empty space as part of the volume

1. Implement eigenvalue ordering and handling. Do not assume that eigenvalues are in ascending order.
2. The image is 8-bits so, we do not need large arrays of 64 or 32-bits
3. The result needs to be a npz or npy file that stores only anisotropy (fa, cl, cs, cp) values per voxel, and a main direction from the voxel, like the average direction.
4. Implement this with clear functions in the script, heavily annotated.
5. Do not use any CLI or command lines. This must run self-contained in Pycharm so make sure the script has the path and file name embedded.
6. Use tqdm to track progress. Make sure that printing is not repeated by the number of workers in the parallelised workflow.
7. You can work a functions.py script and then a main.py that defines parameters, file name/path, saving, stats printing. Results need to be shown in napari.
8. The name of the file we are going to use for this exercise is Data/thresholded\_image.tif
9. I am looking for speed and a small output file. Track overall time from image load onto memory until image is processed. Do not save the image until I close napari.