More details of SCU-Net can be found in this paper: **Jin Z, Xiang Q‐S. Improving accelerated MRI by deep learning with sparsified complex data. *Magn Reson Med*. 2022;1‐14. doi:10.1002/mrm.29556**

One raw dataset downloaded from fastMRI was provide in RawData directory, for demonstration purpose.

The codes include two parts,

1. Codes in Matlab: data preprocessing, complex difference transform, and inverse filtering with data consistency.
2. Codes in Pytorch, SCUNET, deep leaning for sparsified complex data.

Example:

1. First run Readmulticoildata.m file to obtain multi-coil images with 320×320×16 format.
2. Then run ProducePMRIdata.m file to obtain sparsified data for training, validation, and testing.
3. Third, training & validation (run Train.py), and testing (run Test.py).
4. Finally, run Recon\_SCUNET\_PMRI.m file for inverse filtering the sparsified image predicted from CU-Net, and applying data consistency before final reconstruction. The final reconstructed images can be found in \Matlab\FinalRecon directory.

Matlab files

|  |  |
| --- | --- |
| File name | function |
| Readmulticoildata.m | Read data downloaded from fastMRI |
| ProducePMRIdata.m | Obtain the sparsified data for training |
| ComplexNorm.m | To normalize complex data |
| Recon\_SCUNET\_PMRI.m | Recon for predicted data with data consistence |
| VirtualReferenceCoil.m | Combine the multi-coils complex images to single coil complex image |
| ProduceBrainMask.m | Use for extract brain area for SSIM, PSNR, and NMSE calculation. |

Pytorch files

|  |  |
| --- | --- |
| File name | function |
| CBN.py | Complex batch normalization |
| CConv.py | Complex convolution |
| CMaxPool.py | Complex max pooling |
| CReLu.py | Complex Relu |
| CUNet.py | Complex U-type architecture for CNN |
| CUpsample.py | Complex upsampling |
| Train.py | For training |
| Eval.py | For validation |
| Test.py | Predict images using optimal parameters after training |

Reference:

<https://github.com/wavefrontshaping/complexPyTorch>

<https://github.com/ChihebTrabelsi/deep_complex_networks>

<https://github.com/wavefrontshaping/complexPyTorch>

https: // github.com / fchollet / keras / blob / master / keras / layers / normalization.py