REDME: Accelerated MRI KIKI-net

Our goal is to implement KIKI-net on fastMRI data found in the databases from the servers.
We used the Single- Coil dataset that includes knee scans. The dataset is consisting of HDF5 files that contain the following tensors: kspace, reconstruction_rss, reconstruction_esc. We used only the kspace tensor, which is an emulated single-coil k-space data. The shape of the kspace tensor is (number of slices, height, width).

To train the model run the script **model_train.py**
To evaluate a model run the script **model_evaluation.py**

Documentation:

1. **model_train.py** – to train the model, run this script. Gets the defined arguments using them loads the data, sets the model, optimizer, and loss. Finally, train on train data and validation data.
2. **fftc.py** – modified functions for applying FFT, IFFT, and abs on the images.
3. **subsample.py** – creates two types of masks for undersampling the k-space: random and equispaced (explanation in the code). We used equispaced sampling.
4. **transforms_kspace.py** – class KIKIDataTransform processes a raw k-space using the relevant functions: crops the images in the image domain to the required dimension, undersamples the cropped k-space using a mask, returns to dataloader a dictionary
5. **mri_data_kspace.py** – class SliceDataset defines the amount of data to use and load every slice from the h5 files (create the dataset).
6. **DataLoader_kspace.py** – defines the required arguments (def build-args) and creates a dataloader (def DataLoader) using mri_data_kspace.py.
   In this file we changed the hyper parameters. k, i represent the
7. **fastmri_dirs.yaml** – define the path for the data, called in def build-args (it's one of the args).
8. **models.py** – class KIKI is the original model, and class KIKI_res is our adaptation for the model, i.e., adding additional skip connection to the K-net.
9. **layer_utils.py** – the blocks that creates the model.
10. **metrics.py** – functions for model evaluations of the following matrics: MSE, PSNR, NMSE, SSIM.
11. **model_evaluation.py** – To evaluate a trained model, change the models_path to the desired path and run this script. You will get the mentioned metrics and figures of the results.

In order to regenerate our results the parameters which should be defined in DataLoader_kspace.py :

Experiment 1: accelerations = 3, center_fractions = 8/75

Experiment 2: accelerations = 4, center_fractions = 0.08 + comment line 57 and uncomment lines 58 in train_model.py

Experiment 3: k,i = 35 (number of layers), accelerations = 4, center_fractions = 0.08

Experiment 4: accelerations = 2, center_fractions = 0.16 + comment line 62 and uncomment lines 63 in train_model.py

*We tried to tune more hyper params such as batch size, learning rate, lambda but it didn’t improve the results.