In this experiment, we try to construct a CNN network by ourselves. There are four steps of implementation which are data processing, the construction of neural network, training and testing model.

Firstly, for data processing, we need to change the dimension of input data. The initial input and target data shape is (1, 640, 372, 2). So in data processing, the first step is that we compute absolute value to get a real image and the result of shape is (1, 640, 372). Secondly using unsqueeze function to raise dimension to four as (1, 1, 640, 372) because the upsample function in model requires a 4 dimensions input, Finally, the shape become (1, 1, 320, 320) after using T.center\_crop to crop the images to the central 320x320 pixel region.

Then for the neural network model, it contains two convolution layers, two pooling layers, upsampling, and a 1$/times$1 convolution layer and ReLU activation function. The kernels of first and second convolution layer are both 5$/times$5，the stride and padding are both 1 and 2 separately and the out\_channels is 16 and 32 respectively. The two pooling layers are both 2$/times$2. For upsampling, the scale is 4 and mode is bilinear and each layer contains a ReLU. First of all, the input data size is (1, 1, 320, 320), after the first convolution, I extracted 16 features, the size change to (1, 16, 320, 320), because (320+22-5)/1+1=320. Then passing the max pooling, the size reduces to (1, 16, 160, 160), the same with the last layer, the size continues to change to (1, 32, 80, 80) after the second convolution and max pooling. Then we need to use upsampling to improve size to (1, 32, 320, 320) and at last change to (1, 1, 320, 320) by 1$/times$1 convolution layer.

Turn to train the model, I used the dataloader function from PyTorch. Because the size of the input and target data is the same, so the loss function we can choose could be l1 or l2(MSE). Firstly I choose l1 and SDG for the optimizer. I set the learning rate 0.001 after data processing like before and begin to train. Because of the limit of time and the GPU of my laptop, I just trained 2 epochs and save the model. However, I observe the loss has a remarkable fluctuation and keep unchanged in the end.

Fig.1. The reduction of loss curve

By testing the model, the performance of my model is not good. I input the testing data, set AF = 8 and use ssim function test, the average ssim is just 0.45 and the image is below. So I try to modify the network and adjust hyperparameter to optimize the model. Firstly, I change the loss function to MSE and reduce the learning rate to 0.0001. Besides them, I also add a convolution layer to extract 64 features and two 1$/times$1 convolutions to reduce size progressively. After training 2 epochs and testing, I find the performance is still not good, but I do not have enough to continue to adjust the model, and we decide to use U-NET model. Fig.2. The left is the image with undersampling rate 8, the center is the target image and the right is the image after inputting in model.

For dataloader, firstly use load\_data\_path function to load all file names, paths, and slices. Next getting the dataset by MRIDataset function, finally use DataLoader function to get a data iterator which can iterate each set of data. I would mainly introduce the get\_epoch\_batch function in MRIDataset. It could randomly select a few slices from each volume. Firstly it loads the data from the file and transforms to tensor, the apply random mask by MaskFunc function. At last, after undersampling and normalizing data, we got the input data.