Neural Computation Source Code and Report

Report

Group Member Name: Zhangda Xu, Xiaoyu Xia, Lizhao Zhou, Zhun Wang, Wenzhou Liu

Introduction

Now the long acquisition time in MRI often exceeds half an hour which results in low patient throughput. Hence, normal MRI with long acquisition time increases the exam costs and has lots of problems with patient comfort and compliance. People gradually pay more and more attention on safety problems caused by its long duration in MRI machines. Some acute diseases such as acute ischemic stroke, acute intracranial hemorrhage, acute abdominal pain requires the treatment measures can provide fast and valid judgement on the state of illness since long diagnosis time will deteriorate the patients' condition and lead to secondary injury.

In our project, dataset is first preprocessed to get some values for later measurements such as ground truth. After preprocess, we concentrate on undersampling methods called Cartesian undersampling trajectory, respectively 4-fold acceleration and 8-fold acceleration, to accelerate the process of MRI scanning, which is the main process to reduce the scanning time significantly from more than half an hour to only a few minustes. However, after undersampling image quality will be affected, therefore, we try two models CNN and U-Net to reconstruct the undersampled MRI data to obtain more clear images. In result, U-Net deep learning model is more suitable to reconstruct the undersampled images, which will be used as our final results.

Design

Firstly, we load the data by using data_loader.ipynb, and then we show slices in the sample as k-space data and transform k-space data to real images and ground truth. K-space is an abstract space (three-dimensional space) or a plane (two-dimensional space). As MR imaging data is arranged at a specific K-space position according to different spatial frequencies, and finally transformed into an image, so all we need to do is to transform k-space data to real images and compare the original image and the output image by SSIM. Explicitly, K-space uses spatial frequency as the unit (Hz / cm), the spatial frequency K is described by the three mutually perpendicular components Kx, Ky, and Kz. These three vectors correspond to a three-dimensional frequency space, so this abstract space is called K-space.

Then, we will train the neural network model by U-net based on CNN. When it comes to classification, the information provided by the pixels is always taken into account. However, this information generally includes two types: one is environmental field information, and the other is detailed information. The pixel-based approach has a great deal of uncertainty about the choice of the form. Choosing a size that is too large not only requires more pooling layers to make the environmental information appear, but also loses the local detailed information. But U-net uses a network structure that includes down sampling and up sampling. Down sampling is used to gradually display the environmental information, and the process of up sampling is to combine the down sampling information of each layer and up sampling input information to restore the detailed information, and gradually restore the image accurately. Besides, U-Net combines the location information from the down sampling path to finally obtain a general information combining localization and context, which is necessary to predict a good segmentation map.

For model trained by CNN-convolutional neural network-it is mainly used for image recognition and classification. It consists of input layer, convolution layer, pooling layer, fully connected layer (Affline layer), and Softmax layer. There is also a very important structure in convolutional neural networks: filters, which act between layers (convolution layers and pooling layers) and determine how to convolve and pool data.

For Loss Function, we will add a function calculating MSE (Mean Square Error) to give our model feedbacks to improve the performance in reconstruction.

$$MSE(y, y') = \frac{\sum_{i=1}^{n} (y_i - y_i')^2}{n}$$

After gaining the output images, we can transform them to ground truth to make a comparison. And the value range of SSIM is [0,1], which means the larger the value is, the better the performance is. Structural similarity (SSIM) is also a full-reference image quality evaluation index. It measures the similarity of two images from the aspects of brightness, contrast, and structure. The SSIM algorithm is designed to take into account the visual characteristics of the human eye and it is more in line with the human eye's visual perception than traditional methods. MSE or PSNR algorithms are both evaluations of absolute errors. For the fuzzy changes in the human's perception of the structural information of the image, the model also introduces some perception phenomena related to the changes in perception, including the brightness mask and the contrast mask. The structural information refers to the internal dependency between pixels, especially amongst pixels that are close in space. These dependencies carry important information on the target's visual perception.

Implementation

CNN Model

First of all, we try to construct a CNN network by ourselves. There are four steps of the CNN part, which are data processing, the construction of network, training model, 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, firstly we compute absolute value to get a real image and the shape is (1, 640, 372) now. Secondly use unsqueeze function to raise dimension to (1, 1, 640, 372) because the upsample function in model require a 4 dimensions input, Finally use T.center_crop to crop the images to the central 320x320 pixel region(1, 1, 320, 320).

Then for network model, it contains two convolution layers, two pooling layers, upsample and a 1*1 convolution layer and ReLU activation function. The kernel of first and second convolution layer are both 5*5*, the stride and padding are both 1 and 2 separately, and the out_channels is 16 and 32 respectively. And the two pooling layers are both 2*2 of maxpooling. For upsample, 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+2*2-5)/1+1=320. Then passing the maxpooling, the size reduce to (1, 16, 160, 160), the same with last layer, the size continue to change to (1, 32, 80, 80) after the second convolution and maxpooling. Then we need to use upsample to inprove size to (1, 32, 320, 320) and at last change to (1, 1, 320, 320) by 1*1 convolution layer.

Turn to training model, I used the dataloader function from pytorch. Because the size of input and target data is same, so the loss function we can choose could be l1 or l2(MSE), firstly i choose l1 and sdg for optimizer. I set learning rate 0.001, after data processing like before and begin to train. because 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.

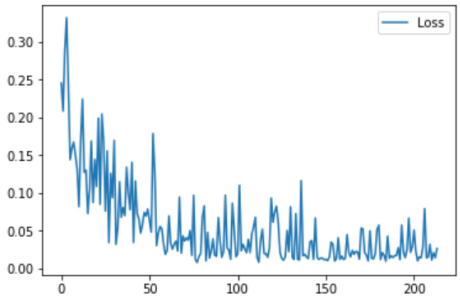

Fig.1. The reduction of loss curve

By testing model, the performance of my model is not good. we 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 we try to modify the network and adjust hyperparameter to optimize the model. Firstly, we change the loss function to mse and reduce learning rate to 0.0001. Besides them, we also add a convolution layer to extract 64 features, and add 2 1*1 convolutions to reduce size progressively. After training 2 epochs and testing, we find the performance is still not good, but we do not have enough time to continue to adjust model, and we decide to use unet model.

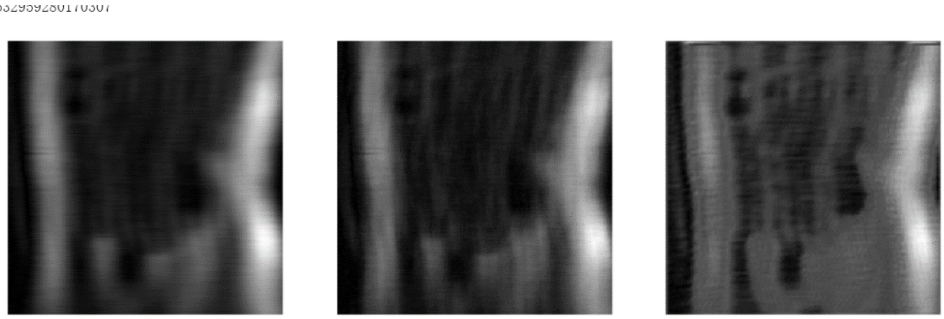

Fig.2. the left is the image with undersampling rate 8, the center is the target image and the right is the image after inputting in model.

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. We would mainly introduce the get_epoch_batch function in MRIDataset. It could random select a few slices from each volume. Firstly it loads the data from file and transform to tensor, the apply random mask by MaskFunc function. At last after undersampling and normalizing data, we got the input data.

U-Net Model

Experiments

Conclusion

Description of Contribution

In [ ]: