Coursework 2019/2020

Released: Wednesday November 13th 2019
Deadline: Friday December 13th 2019 at 23:59
Weight: 20 %

The module is assessed by 80% examination and 20% continuous assessment. This document specifies the continuous assessment component. The objective of this coursework is for you to gain practical experience in designing, implementing, training and optimizing a neural network to carry out a specific real-world task. The coursework is designed as a competition where you will work together in groups, each group submitting a solution. The allocation of students to groups and the announcement of competition results will be specified on Canvas.

All files must be submitted via Canvas within the deadline by the group leader. If you miss this deadline, your mark will be reduced by 5% (out of 100%) for each School working day (or part thereof) your submission is late. Feedback with marks will be returned within three weeks of the hand-in deadline.

1. Introduction

The task of this year is to reconstruct images using *magnetic resonance imaging*[1] (MRI), which is a very powerful diagnostic tool for a wide range of disorders, including neurological, musculoskeletal, and oncological diseases. However, the long acquisition time in MRI, which can easily exceed half an hour, leads to low patient throughput, problems with patient comfort and compliance, artefacts from patient motion, and high exam costs. With modern machinary algorithms, this coursework concerns how fast we can push to reconstruct high-quality MRI images.

First, let us familiarise ourselves with the MRI reconstruction problem[2]. The data acquired from a MRI scanner is a complex-valued k-space[3] data, $y \in \mathbb{C}^N$, where k-space represents the spatial frequency information (Fourier coefficients) of an object imaged. To reconstruct that image, one simply performs the inverse *discrete Fourier transform*[4] (DFT) to the k-space data

$$m = \mathcal{F}^{-1}(y), \tag{1}$$

where $m \in \mathbb{C}^N$ is the resulting complex-valued image and $\mathcal{F}^{-1}$ denotes the DFT. Figure 1 presents you four examples of such a mapping from the k-space data to the image space. Figure 1 top shows you the fully sampled cases, where the MRI scanner sampled k-space over all the Cartesian grid.

[1] If you are interested in some MRI basics please read http://mri-q.com/hellipmagnets--scanners.html

[2] In addition to MRI reconstruction, in many other image reconstruction problems one usually needs to bring a raw data acquired from some modality to an image-like data using certain inverse spatial transform. These processes are typically governed by Fredholm integral equations.

[3] See http://mriquestions.com/what-is-k-space.html and https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3097694/pdf/biij-04-e15.pdf for more details.

[4] https://medium.com/sho-jp/fourier-transform-101-part-4-discrete-fourier-transform-8fc3fbb763f3.

This sampling method has the advantage of reconstructing images with great details, as shown in Figure 1 bottom. However, it takes the MRI scanner relatively long time to acquire these fully sampled k-space data. To reduce scanning time, undersampling strategies are often used.

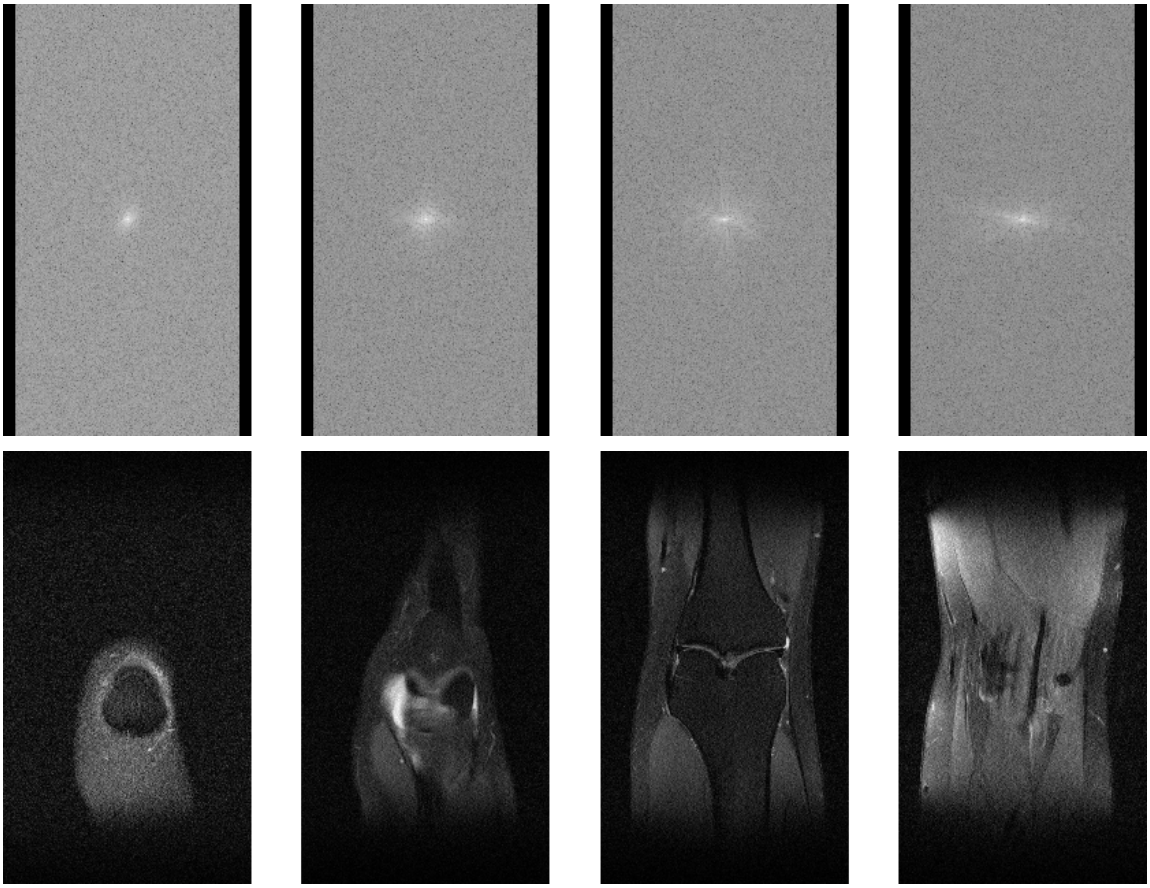

Figure 1: K-space data (top) versus image space data (bottom). The relation between k-space data and image data is a discrete Fourier transform, as shown in Eq (1). Images here show a human knee at different anatomical positions.

We introduce one simple undersampling strategy here - Cartesian undersampling trajectory[5]. First, we generate two random mask functions using such a trajectory, as shown in Figure 2. For both masks, the entries are 1 on the white strips and 0 on the black background. If the original k-space data is multiplied by the left mask, as shown in the top row of Figure 3, we end up only using 1/4 of original k-space data. In this case, the acquisition speed is 4 times faster. If we use the mask on the right in Figure 2, we can achieve an acquisition speed 8 times faster. In the bottom row of Figure 3, we displayed the corresponding k-space data masked using the right mask in Figure 2.

[5]If you are interested in other sampling strategies, please read Image Acquisition Section on https://people.eecs.berkeley.edu/~mlustig/CS/CSMRI.pdf for more details.

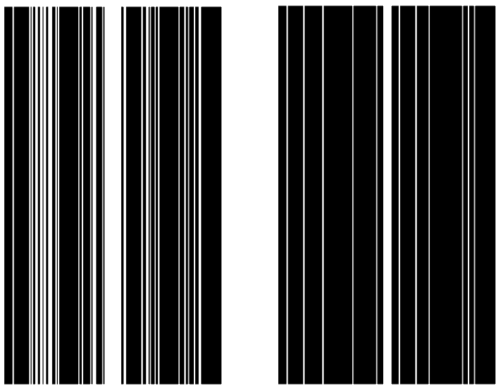

Figure 2: Examples of undersampled k-space trajectories. Left: 4-fold acceleration; Right: 8-fold acceleration.

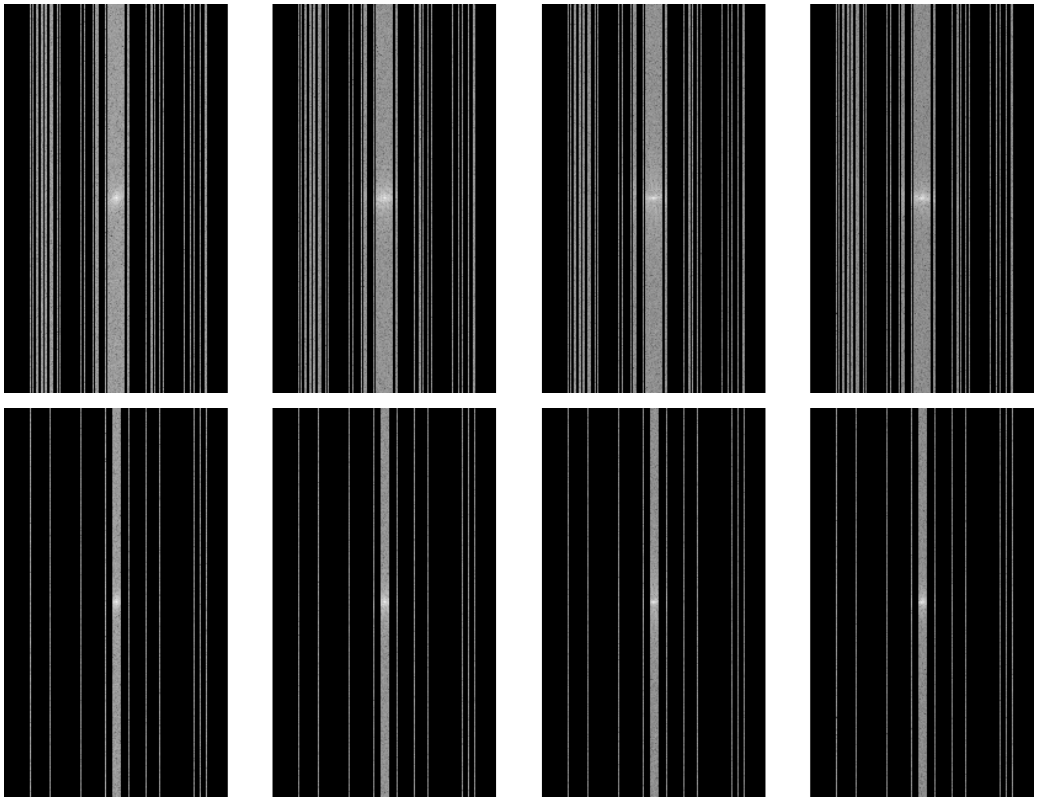

Figure 3: Masked k-space data. Top: the results after multiplying the 4-fold acceleration mask with each of four k-space data in Figure 1 top. Bottom: the results after multiplying the 8-fold acceleration mask with each of four k-space data in Figure 1 top.

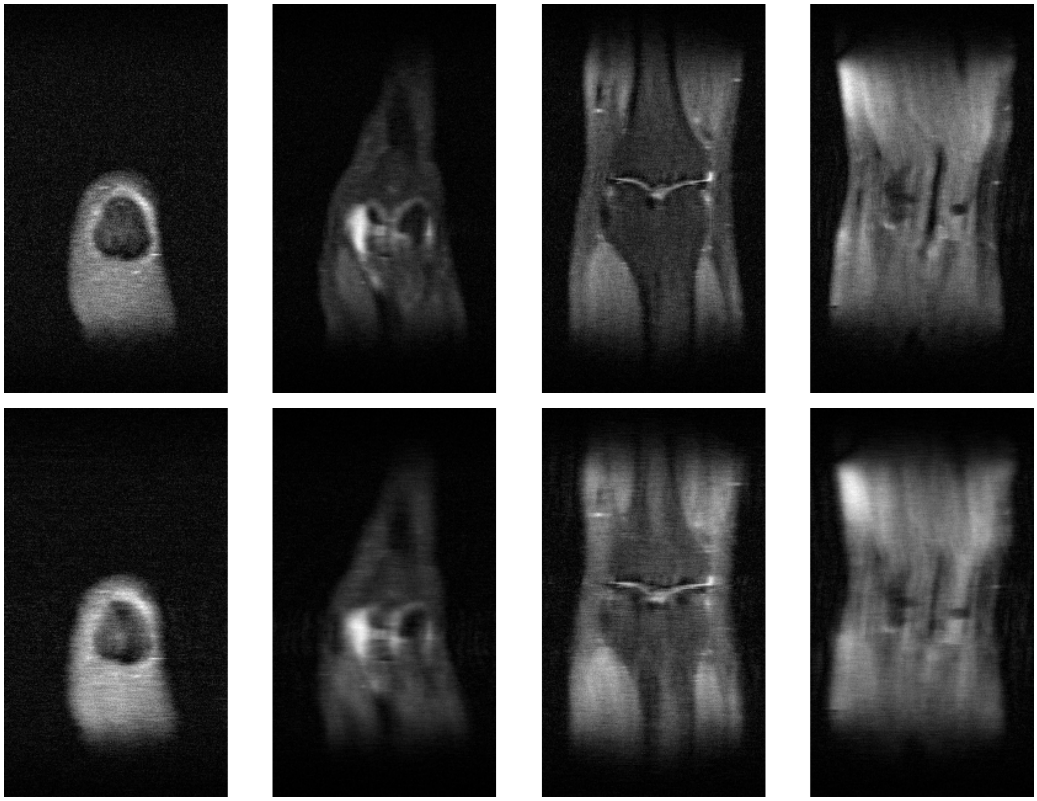

Figure 4: Reconstruction from undersampled k-space. Top: the results after applying the inverse DFT to the k-space data in Figure 3 top. Bottom: the results after applying the inverse DFT to the k-space data in Figure 3 bottom.

Let us now see what the images we are able to reconstruct by applying Eq (1) to the undersampled, masked k-space data in Figure 3. As shown above, as the undersampled rate, also known as the acceleration rate (AF), goes higher, the reconstructed images become more blurred. However, the k-space data producing images at the bottom of Figure 4 can be very quickly acquired from a MRI scanner. The speed is 8 times faster than we acquire the fully sampled k-space data that produce the images in Figure 1.

Ground truth: To train a deep learning model with our MRI dataset, we define the ground truth image, which is the absolute value of the complex-valued fully sampled image, calculated (via inverse DFT) from the complex-valued fully sampled k-space. Note that the ground truth images are real-valued images. Also note that all ground truth images are cropped to the central 320×320 pixel region to focus on the region of interest, where the final evaluation is carried out. Note: ***You should never crop raw k-space data, as all data points in k-space contribute to the image content.***

Metrics: The *structural similarity*[6] (SSIM) index will be used to measure the reconstruction per-

[6] https://en.wikipedia.org/wiki/Structural_similarity

formance with respect to ground truth. SSIM measures the similarity between two images by exploiting the inter-dependencies among nearby pixels. It is inherently able to evaluate structural properties of the objects in an image and is computed at different image locations. We will provide the SSIM function for you to measure the success. Note: ***our SSIM implementation is computed based on three-dimensional volumes rather than two-dimensional slices. If you have 10 volumes in the test set, you will compute 10 SSIM values***.

Loss function: As a good and easy start point, we recommend to use the L1 loss to train your network for the MRI dataset. It is defined as

$$L_1(\widehat{v}, v) = \|\widehat{v} - v\|_1, \tag{2}$$

where $\widehat{v}$ is the network output and v is the corresponding ground truth. You are also encourage to use other loss function, including the SSIM loss, the VGG loss, the GAN loss, the L2 (MSE) loss, or a combination of some of them.

Dataset: In this coursework, we use a subset MRI data from the fastMRI[7] challenge. Our MRI dataset contains raw k-space data from 100 three-dimensional volumes, each having about 30-40 two-dimensional slices. We split the data into 70% for training and 30% for testing. For the training set, each volume contains only fully sampled k-space data, where you can compute ground truth images as well as simulate undersampled data (either 4-fold AF or 8-fold AF) by using the random mask generator we provided. For the test set, each data contains a 4-fold undersampled k-space data and a 4-fold mask that generates the undersampled k-space. The 8-fold undersampled k-space data and the 8-fold mask were also included. You may want to use the masks provided to develop more advanced learning methods. Note that for the test set, we hold the fully sampled k-space data, where we can easily compute the ground truth images. ***Your mission is to see how accurate your method can reconstruct an image from the corresponding undersampled k-space for both undersampled rates, by comparing your reconstruction with the ground truth image we hold***.

2. What you need to do

1. Log in a computer in LG04 and open the terminal. Download the datasets and tutorials by typing in (it takes a while to copy over all data, so be patient)

   ▷ `cp -r /home/staff/duanj/NC2019MRI /tmp`
   ▷ `module load neural-comp`
   ▷ `jupyter-notebook`

2. Once you open `jupyter-notebook`, go to `/tmp/NC2019MRI/fastMRI/MRI_tutorial.ipynb`. Make sure you go through the tutorial carefully in order to better understand the problem. In addition, we also provide a data loader in `/tmp/NC2019MRI/fastMRI/data_loader.ipynb` to ease the difficulty of the problem so you can focus on only methodology development.

3. Design a suitable neural network that reconstructs images with good quality for both training

[7] https://fastmri.org/

and test sets. This method should be implemented and tested for both 4-fold and 8-fold acceleration rates.

4. Upload your reconstruction results to either Google drive or Dropbox at a specific time and inform us the link. We will download, evaluate and rank your reconstructions. The ranking will be made available via Canvas.

5. Write one group report explaining what you did and what you found. A reasonable length for the report would be between 3000 and 4000 words excluding the reference list, plus as many diagrams, tables and graphs as you think appropriate. You should include the following sections:

    - ▷ Introduction - Discuss the data sets involved, the machine learning task, and what you aimed to achieve. [5%]
    - ▷ Design - Describe and justify the neural network you designed for the task, and the factors you decided to experiment with. [15%]
    - ▷ Implementation - Describe how you implemented your neural network and the associated performance analysis mechanisms. Explain why you chose to do it that way. Remember to cite any sources you used. Include comments in the source code that explains how the code works. [20%]
    - ▷ Experiments - Describe the experiments you carried out to optimise your network's generalization performance, and present the results you obtained. Explain in detail how you used the training and test data sets. The results should be presented in a statistically rigorous manner. [45%]
    - ▷ Conclusions - Summarize your key findings, including which factors proved most crucial, and what was the best generalization performance you achieved. [10%]
    - ▷ Description of contribution - Describe each group members' contribution to the overall project. [5%]

3. Files to Submit

The group leader should submit on Canvas a ZIP archive containing the following two files, where `gn` should be replaced by the group number:

- ▷ `gn-coursework.ipynb`
  Source code and report in the form of an executable Jupyter notebook. The Jupyter notebook should have the section headings as described above. At the beginning of the notebook, please include a web link (either Google drive or Dropbox), where we can download your reconstruction results for the test data set.
- ▷ `gn-coursework.pdf`
  A PDF version of the Jupyter notebook file.

4. Assessment

The groups will be marked by their report and source code as indicated by the percentages above. The outcome of the competition will not impact the assessment. If there is evidence of significant differences in contributions among group members, then individual marks may be adjusted.

Keep in mind that this coursework corresponds to only 20% of a 20 credit module. Spend an appropriate amount of time on it!

5. Help and Advice

If you get stuck, feel free to ask for advice during Jinming Duan's office hours which are at 3-5 PM every Friday during Autumn term in Room 108 in the School of Computer Science. You can also write questions and comments in the Discussion forum on Canvas.

It can be challenging to find suitable time and place for all group members to meet. It is therefore highly recommended that you make use of online collaborative tools. In particular, you could set up a discussion channel for the project on Slack[8], and keep the code in a git repository, e.g. on the School of Computer Science's git server[9], or with some external provider such as Bitbucket or GitHub. You can also arrange virtual meetings via Google Hangout.

[8] https://slack.com
[9] https://git-teaching.cs.bham.ac.uk/