MRI Reconstruction

Submitted as final project report for Deep Learning, IDC, 2019

1 authors

Nimrod Feldman (ID. 311129555) Ron Maayan (ID. 311125652) Segev Efraim (ID. 311486773)

2 Introduction

MRI images with high resolution are expensive and time consuming. Patients' bodies have to be exposed a long period inside the MRI scanner. In order to reduce the time of scanning, the MRI scanner is commanded to avoid some frequencies while performing the scan. However, this leads to poor the quality of the MRI images at the end of the process.

2.1 Related Works

Please provide some references to existing works, if you found any. it's an optional sub section, but can definitely help.

3 Solution

3.1 General approach

We want the MRI scanner to continue avoiding some frequencies to reduce the time of scanning. However, to counteract the lower resolution we aim to use neural networks to improve image quality (using the low quality MRI images as input).

In order to do this, first, we need to design a neural network which will learn to reconstruct the low resolution MRI images. To design it, we need to agree with a dataset structure. The dataset was provided by NYU and can be found in https://fastmri.med.nyu.edu. It is composed by two sets: one for training 70% and other for testing 30%.

More technically, we have two kinds of undersampled k-space data in each dataset sample: 4-fold undersampled k-space data and 8-fold undersampled k-space data. 4-fold undersampled k-space data contains more information about

frecuencies of the scanned body part than 8-fold undersampled but they both generate low resolution images. MRI images originated from 8-fold have lower resolution than 4-fold.

Our objective it's to craft a neural network model that can reconstruct MRI images originated from 4-fold k-space samples and 8-fold k-space samples in the provided dataset. In order to mesure the performance of the neural network model we will use MSE (Minimal Squared Error) between the predicted image (output of the model) and its respective high resolution version.

3.2 Design

We used Pytorch library to implement our model.
As we learned in class convolutional network models (CNN) are widely use to treat images in neural networks. The convolution operation in 2D is simple but very useful to capture the features or patterns of an image.

Since undersampled and fully sampled MRI images in our dataset are composed by two channels, we start the convolutional neural network with 2 channels in input. The number of channels is normally incremented after each operation, these new channels could be seen as the different "views" of the same image.

There is a similar study about reconstructing MRI images from undersampled k-space data in the web.
Max pooling : Goes trough the view with a kernel matrix to output the max value in the kernel.
Avg unpooling : to recover the size of the view (reduced previously by the max pooling operation).
Copy and concat : in order to localize features more precisely after recovering the size.
ReLU : discriminative function, output from 0 to 1.

The process we have taken as reference is the U-net We transform the undersampled k-space sample into a MRI image. Then, the model apply convolution and ReLU to the MRI image to obtain 64 normalised channels that have captured the image's features. After this, we do a max pool operation, so the size of the views are reduced but features are emphasized. The process goes fowards similarly adding more channels and reducing the size of the views. Later, we recover gradually the size of the views with the avg unpool operation, also, we concatenate previous channels in oder to localize features more precisely and then apply convolution-ReLU to reduce those channels. At the end of the process, we apply a final convolution to obtain the reconstructed MRI image.

We apply a correction of the k-space at the output of the model to gain more resolution. In order to do this, the output MRI image is converted into k-space then it is partially corrected with the input k-space of the model (the input content is valuable) and then reconverted to a MRI image, improving the results. We will evaluate the improvement of the results when applying the k-space correction with SSIM in the experiments part of this report.

4 Experimental results

4.1 Training set, validation set and test set

We have splitted the training set in two parts: a training set(70%) for training the model and a validation set for validating the model(30%). We validate a model focusing on the loss function, in the training process we reduce the loss function at each epoch. In the meanwhile, we need to know when to stop to avoid overfitting. To solve this problem, we stop the training when results of the model in the validation set don't improve anymore or conversely they start to deteriorate. We will only use the test set to measure the results because the test set needs to be totally isolated of the model training and evaluation. We had some piratical and theoretical challenges. Understanding the problem, how MRI machines works, k-space and "forier transformation" are new concept for us. Our major technical challenge were different sizes of images in the dataset. We had to cut images and loss information to reach equivalent size for all.

4.2 Evaluation

We will evaluate the improvement of the results when applying the k-space correction with SSIM in the experiments part of this report. In order to improve the results, we have used a variant of the u-net neural network: the dense u-net approach. In a dense u-net, we can add more layers inside the u-net process in which we apply convolutions which keep the size of the views and the number of channels.

4.2.1 Dense 4-fold

epoches	3
learning rate	1e-3
weight decay	0
Average SSIM score	0.6413

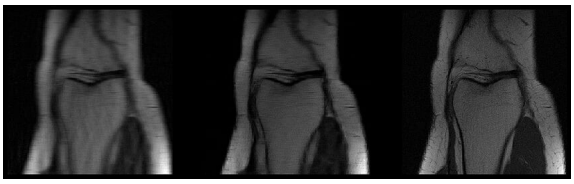


Figure 1: The results could be perceived in the figure. On the left, we have the low resolution MRI image, in the middle the prediction of our model and on the right we have the high resolution image taken as reference.

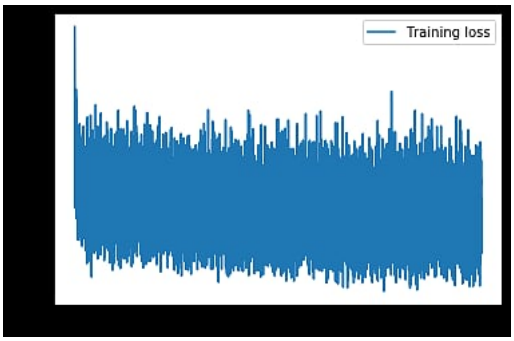
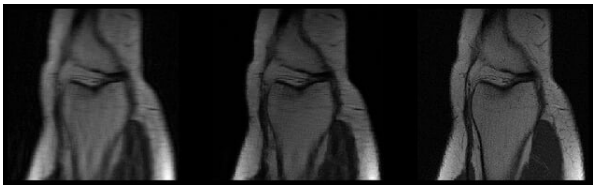


Figure 2: We can see also how the loss is decreasing in the training.

4.2.2 Dense early stop 4-fold

epoches	3
learning rate	1e-4
weight decay	1e-2
Average SSIM score	0.6319



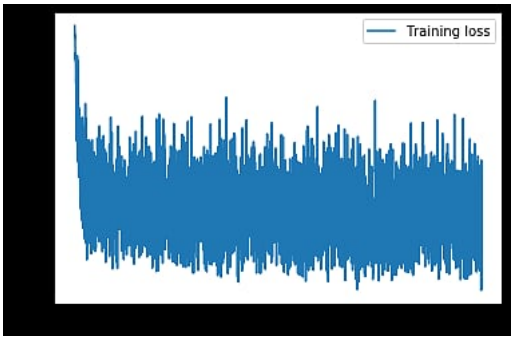


Figure 3: We can see also how the loss is decreasing in the training.

4.2.3 Dense early stop 8-fold

epoches	3
learning rate	1e-3
weight decay	0
Average SSIM score	0.5185

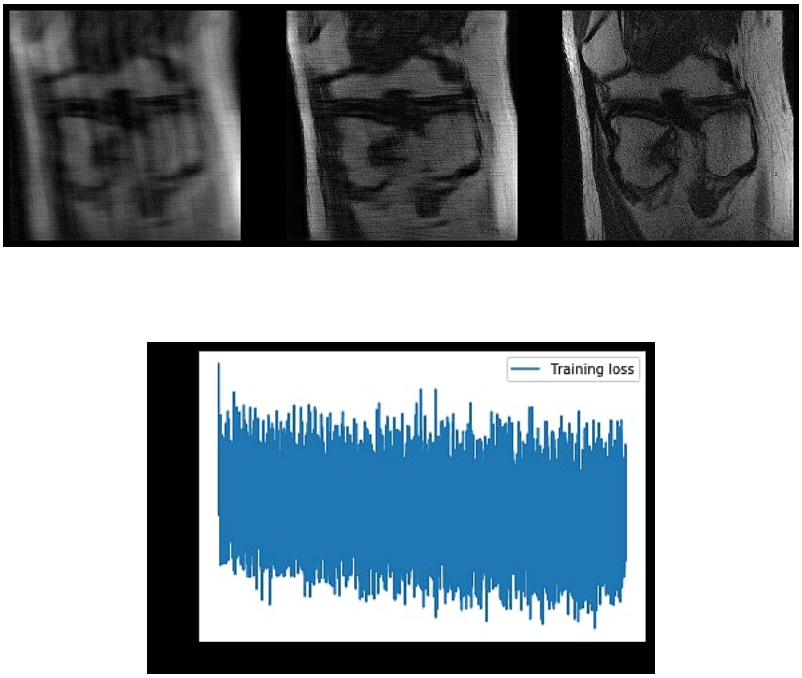


Figure 4: We can see also how the loss is decreasing in the training.

4.2.4 Downsampling 4-fold

epoches	3
learning rate	1e-3
weight decay	0
Average SSIM score	0.6420

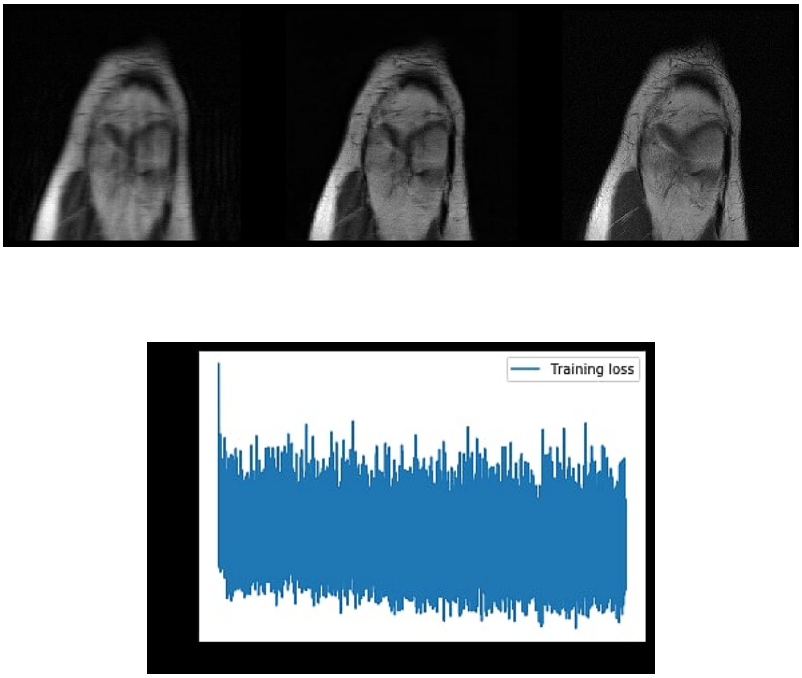


Figure 5: We can see also how the loss is decreasing in the training.

4.2.5 U-net 8-af batch k-space correction

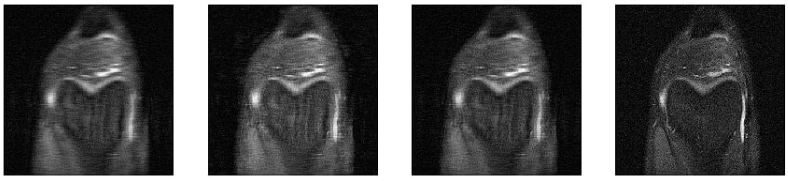


Figure 6: On the left we have the undersampled MRI image, the second one is the prediction of the model, the third one is the result after applying k-space correction and the final one on the right the fully sampled MRI image as reference.

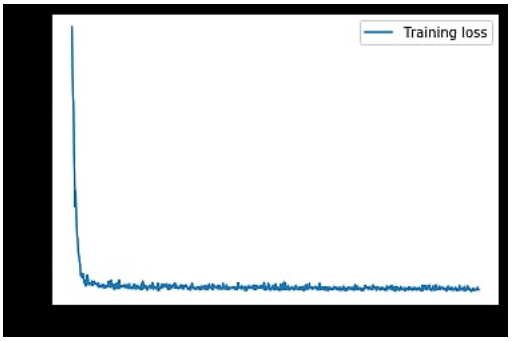


Figure 7: For this U-net 8-af model, it can be clearly presented that the loss of this training tends to zero which has the best performance almost these models tested before.

name	epochs	learning rate	weight decay	Avg SSM
denser-4af	3	1e-3	0	0.6413
denser-4af	3	1e-4	1e-2	0.6319
denser-8af	3	1e-3	0	0.5185
downsampling-4af	3	1e-3	0	0.6420

5 Discussion

Reconstructing MRI images with U-net neural networks showed promising results. We can see the network provided images that looks similar to the high quality images. Since small details are extremely important when deciphering MRI images, we are not sure this solution can be used by industrial entities. For further improvement we suggest 2 improvements. First, passing as input to the network the k-space images and not the images constructed by "forier transformation". Second, more neurons in the deep layers of the network to learn detailed information about the bones structure.

6 Code

https://github.com/nimrod4278/nn-final-proj.git

References

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6785508/
https://github.com/hpkim0512/Deep$_M RI_U net/$
$https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/$
$https://github.com/usuyama/pytorch-unet/$