CS 7643: Fast MRI

Enrico Zerilli
Georgia Institute of Technology
ezerilli@gatech.edu

Sudipto Lodh
Georgia Institute of Technology
slodh3@gatech.edu

Robert Bartel
Georgia Institute of Technology
rbartel3@gatech.edu

Abstract

Magnetic Resonance Imaging is a fundamental diagnostic tool for a large spectrum of disorders, including oncological, musculoskeletal and neurological diseases. Unfortunately, its major weakness is its long acquisition time, which easily exceeds 30 minutes. This poses several limitations to its applicability, ranging from its high costs to common motion artifacts. Thus, accelerating its acquisition speed is critical to fully take advantage of its considerable benefits, mainly the absence of harmful X-rays and its excellent soft tissue contrast. Current techniques use parallel imaging through multiple coils and compressed sensing to acquire fewer measurements. However, more accurate reconstruction can be achieved via the use of Deep Learning on large datasets, that includes k-space data, like the NYU fastMRI dataset [9]. In this project, we show how self-supervision can be beneficially used with comparable results to the supervised case, without the need to acquired prohibitively expensive fully-sampled data.

1. Introduction/Background/Motivation

Medical diagnosis of a wide range of disorders makes, nowadays, extensive use of imaging techniques such as X-Rays, Ultrasounds, Computed Tomography (CT) or MRI. In particular, *Magnetic Resonance Imaging* has become a powerful and flexible technique that provides accurate details and excellent soft tissue contrast, especially for neurological, musculoskeletal and oncological diseases [9]. MRI consists in emitting a sequence of magnetic fields, varying in time and space, and measuring the resonant electromagnetic response through one or multiple receiver coils for all different frequencies. This allows to fully-sample the interested region in Fourier space (also known as *k-space*) up to a maximum frequency. The spatially-resolved image m can then be reconstructed simply by inverse Discrete Fourier Transform (DFT) of the *k-space* image y:

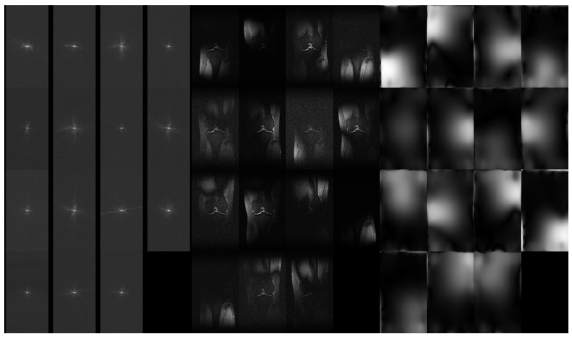
$$\hat{m} = \mathcal{F}^{-1}(y) \tag{1}$$

Nonetheless, the use of MRI as a diagnostic tool is greatly limited by a number of relevant factors, first of all its long acquisition time, which easily exceeds 30 minutes in most of the cases. As a direct consequence, MRI has low patient throughput and high costs [9]. Also, patients are not comfortable in staying perfectly still for such a long time and often cause artifacts from motion, which highly impact the quality of the diagnosis. This makes MRI less suited to several diseases, for which CT is preferable due to its shorter acquisition time and its minor costs. Therefore, increasing acquisition speed has been a major goal of research related to MRI in the last decades. Indeed, a shorter acquisition time would allow to decrease its costs, to enlarge the spectrum of disorders for which MRI can be used, to avoid harmful X-rays to patients through the use of CT, to diminish artifacts from patient motion and improve thus the quality of the acquired images. Unfortunately, two simple accelerations, like sampling up to a lower maximum frequency or undersampling, either decrease the spatial resolution of the reconstructed image or introduce *aliasing* artifacts by violation of the Nyquist-Shannon sampling theorem[1], and are thus not applicable.

Since the advent of MRI in the 1970s, the introduction of *parallel imaging* has improved acquisition times by using multiple coils to acquire several views of the imaged volume in *k-space*, each modulated by a complex-valued position-dependant coil sensitivity to the MR signal [9]. The *k-space* image y_i measured by coil i of n_c is:

$$y_i = \mathcal{F}(S_i m) + noise = g_i * \mathcal{F}(m) + noise \tag{2}$$

[1] the sampling frequency f_s should be at least twice as the maximum frequency in the spectrum f_{max} to avoid aliasing: $f_s > 2f_{max}$.



(a) k-space data from 15 coils (b) Individual coil spatial images from fully sampled data (c) Coil sensitivity map magnitudes given by ESPIRiT

Figure 1. Multi-coil MRI reconstruction [9].

where S_i is the coil sensitivity and g_i its Fourier transform, which can be estimated either by separate low-resolution scans or directly from the *k-space* measurements by a fully-sampled small central (low spatial frequencies) region [9]. This is shown in Figure 1. In the case of a fully-sampled *k-space*, the above represents a set of linear equations that is overdetermined by a factor n_c and can thus be pseudo-inverted to reconstruct m, as long as the system is full-rank. On the other hand, this overdetermination can be exploited to accelerate acquisition by undersampling the *k-space*. As long as there are more measurements than image voxels to be reconstructed, a least-squares solution can be used to get an n_c-fold acceleration. In practice, different factors, such as a tight resolution, spread out the coil sensitivities, thus lowering the rank of the linear system, which leads to the need of regularization and to smaller typical acceleration factors.

Compressed sensing (CS) provides a framework to reconstruct an image whose measured (undersampled) *k-space* y matches closely its Fourier transform, subject to sparsity constraints. As such, the reconstructed image best approximates the ground-truth image that would be reconstructed from fully-sampled data. In the single-coil case, this optimization problem can be expressed in its Lagrangian dual form as:

$$\min_m \frac{1}{2}||\mathcal{P}(\mathcal{F}(m)) - y||_2^2 + \lambda R(m) \tag{3}$$

where, $\mathcal{P}$ is a projection mask that performs undersampling and R is a convex regularizer typically chosen to be either the L1-norm of the image, or the L1-norm of a wavelet representation of the image (which is typically sparse), or a total-variation (TV) penalty penalizing spatial gradients [9]. In the multi-coil case, which provides increased *Signal Noise Ratio* (SNR) over extended fields of view and can take advantage of *parallel imaging* speedup, the CS formulation given by 3 becomes:

	Volumes		Slices	
Coil	Single	Multi	Single	Multi
training	973	973	34,742	34,742
validation	199	199	7,135	7,135
test	118	108	4,032	3,903
challenge	104	92	3,810	3,305

Table 1. Volumes and slices per task and set.

$$\min_m \frac{1}{2}\sum_{i=1}^{n_c} ||\mathcal{P}(\mathcal{F}(S_i m)) - y_i||_2^2 + \lambda R(m) \tag{4}$$

which can be fused with *parallel imaging* in the ESPIRiT approach [7]. Unfortunately, CS has the major drawback of introducing compression artifacts and oversmoothing (lost of details fundamental to radiologists for diagnosis), while being computational expensive. Moreover, CS frequently makes use of hand-crafted features, that are not those representing the actual data.

New emerging approaches attempt to improve these baselines through the use of Deep Learning models. Specifically, this has been enormously facilitated by the publication of the NYU fastMRI dataset [4], whose primary goal is to test whether machine learning can aid in the reconstruction of medical images [3, 9]. This dataset contains a large amount of anonymized MRI scans of knees and brains, providing: raw multi-coil *k-space* data; emulated single-coil (ESC) *k-space* data, that is linear combinations of multi-coil raw data fitted to the ground-truth root-sum-of-squares (RSS) in a least-squares sense; ground-truth images reconstructed from fully-sampled multi-coil acquisitions through a simple RSS of the inverse DFT and, for single-coil, also as inverse DFT of the ESC data; additional DICOM images representing a larger variety of machines and protocols, but for which raw data is not available. The dataset thus allows both single-coil and multi-coil reconstruction tasks from undersampled data. Data is provided separately for each task and already splitted in training and validation sets (containing fully-sampled scans), as well as test and challenge sets (containing undersampled data). Knee multi-coil raw data contains 6970 scans, acquired using 15 coils, matrix size 320x320, slice thickness 3mm and two protocols: a coronal proton-density weighting (PD) with (798 scans) and without (796 scans) fat suppression. Brain multi-coil raw data contains 1594 scans and includes axial T1 weighted (some of which with contrast agent, T1 POST), T2 weighted and FLAIR protocols. In the brain scans, only axial scans up to the orbital rim has been used to ensure data de-identification. Table 1 summarizes the number of volumes and slices per task in the provided sets

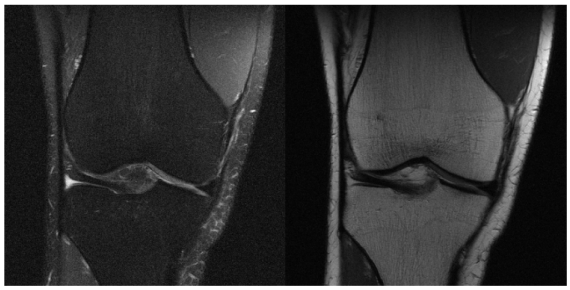


Figure 2. A PD knee image with fat suppression (a) and without (b). Being a soft tissue, fat as a high response in MRI, but makes details difficult to see [9].

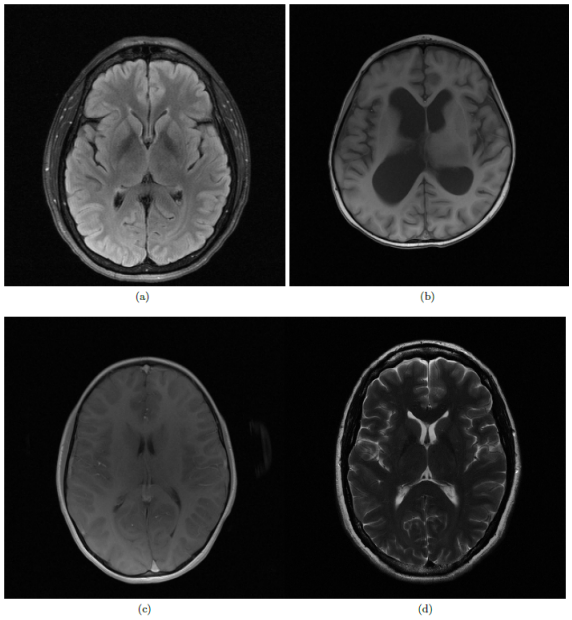


Figure 3. Axial brain slices with FLAIR (a), T1 weighted (b), T1 POST (c) and T2 weighted (d). Each protocol allows to highlight different structures. [9]

(only in the training and validation sets the same slices are shared between the single-coil and multi-coil frameworks), while Figure 2 and 3 show slices of knee and brain for the different protocols used.

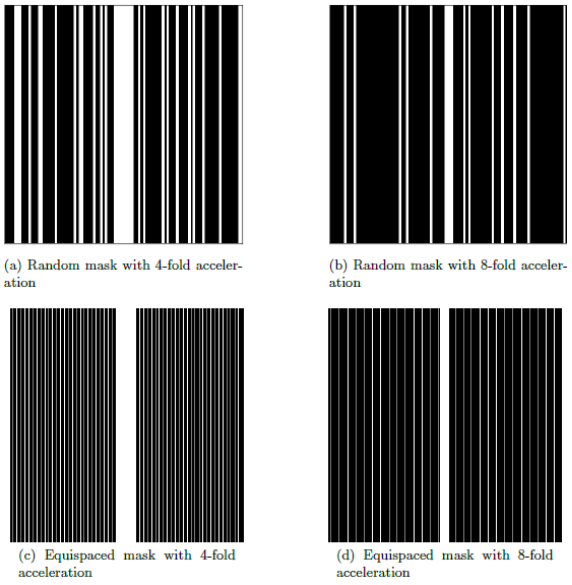
In the context of this project, due to dataset size constraints w.r.t. the available hard-drives, **we are going to limit our study to the case of single-coil knee data, for which sizes are tractable**: training ∼88 GB, validation ∼19 GB, test ∼7 GB, challenge ∼1.5 GB. Unfortunately, the multi-coil data both for knee and brain is just too big for us to use (just the training sets are about 1 TB each). Also, DICOM data will not bee used, since it does not provide with raw *k-space* data.

2. Approach

NYU fastMRI provides a large dataset and a code base [4, 5] that can be leveraged to experiment with Deep Neural Networks (DNNs) in the context of MRI acceleration through subsampling [9]. These models are often trained using supervised methods that are at the moment the current *de facto* choice, when it comes to applying these models in the context of MRI. The simplest of these image reconstruction models are U-Nets [6] and ResNets [2]. More advanced hybrid techniques involving *parallel imaging* data models with DNN-base priors are current state-of-the-art and some of them are even being adopted in the clinic, but we will not explore those.

Rather, our approach consists first in **re-training from scratch the U-Net model** provided in the fastMRI code base [5], that will become our **baseline**. However, supervised learning methods require a prohibitively expensive process of annotating the collected data with ground-truth outputs. This is especially true in the context of MRI, where collecting a large amount of fully-sampled *k-space* data is quite costly and *k-space* acquisitions are naturally subsampled via *parallel imaging*. Therefore, we will concentrate our attention on self-supervised methods and attempt to implement a **self-supervision hold-out method** [8], which have yet to be assessed on a large dataset like fastMRI. Finally, we are going to examine a second self-supervision method, involving **self-distillation with no labels (DINO)** [1]. However, in our case we are performing an image reconstruction (i.e. a pixel-by-pixel regression problem) and not a classification problem. Therefore, the DINO framework needs a lot of adaptations and will be attempted as additional framework.

In all our experiments, performed on the single-coil task, the undersampling is performed by masking *k-space* lines from a fully-sampled acquisition to all slices in a volume along the phase encoding direction, in order to simulate an acceleration via subsampling that is actually realizable in real-world scenarios. For obvious reasons related to keeping most of the low-frequency information contained in a slice, undersampling masks are produced by first cropping a small center region of lowest frequencies in the *k-space*, representing either 8% (4-fold acceleration) or 4% (8-fold acceleration) of all *k-space* lines. The remaining higher frequency *k-space* lines are included uniformly at random for the knee and equidistant for the brain, until the desired acceleration factor is achieved [9]. Examples of canonical undersampling masks are shown in Figure 4. The zero-filled image thus obtained emulates the process of not observing the masked *k-space* lines. Before providing it to the model, the magnitude of its inverse DFT is computed to convert it back to the image space and the result is cropped to 320x320



(a) Random mask with 4-fold acceleration

(b) Random mask with 8-fold acceleration

(c) Equispaced mask with 4-fold acceleration

(d) Equispaced mask with 8-fold acceleration

Figure 4. Examples of undersampling masks. [9]

to remove any readout and phase oversampling [9].

2.1. Re-training from scratch the U-Net model

FastMRI provides with a baseline U-Net model, trained in a supervised fashion, which has successfully been used for image-to-image reconstruction tasks in many fields including MRI [5]. The model is depicted in Figure 5 and consists of two DNNs in an encoder-decoder architecture. The encoder performs the downsampling of the image to a latent space and consists of blocks of two 3x3 convolutions each followed by Instance Normalization and ReLU or Leaky ReLU activations. In between two consecutive blocks, a downsampling operation, consisting of max-pooling or average-pooling is performed with stride 2, which reduces spatial dimensions by half. On the other end, the decoder performs the upsampling of the latent space back to an image space of same dimension as the input and consists of convolution blocks analogous to the downsampling path. However, in the decoder, these blocks are interleaved with upsampling operations consisting of a transpose convolution which doubles the spatial resolution between blocks. Moreover, each of these blocks in the decoder is connected with a skip connection with the output of the convolution block of same dimensions in the encoder. Thus, a convolution block in the decoder, not only inputs the upsampled output activations from the previous block, but also the output activations of the convolution block of same dimensions in the encoder, concatenated along the channel dimension. This is intended to help significantly with training by improving the gradient flow. Finally, at the end of the upsampling path, a series of 1x1 convolutions

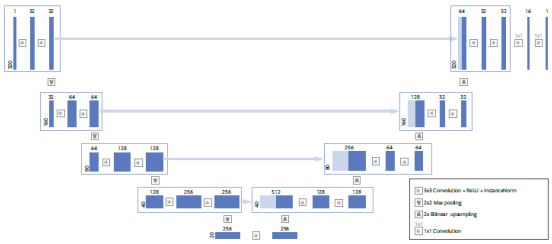


Figure 5. Single-coil baseline U-Net model. [9]

reduces the number of channels back to 1 without changing the spatial resolution [9].

This single-coil U-Net baseline is trained for 50 epochs on a training set of 973 volumes containing 34,742 slices and validated on 199 volumes containing 7,135 slices, as reported in Table 1. In order to do that, we use an RMSProp optimizer with initial learning rate of 0.001, reduced by 10 at epoch 40. We explicitly decide not to perform fine-tuning on the pre-trained U-Net model, as this would not be enough challenging. We did not anticipate any problem with that, considering that the model has been already successfully trained as baseline model by fastMRI. Nonetheless, we encountered a lot of problems with the size of the dataset and the training time, which takes more than **3 days of training (!) despite our Nvidia Quadro P2200 with 64GB of RAM**. This limited enormously the re-training on the U-Net model and consecutively impacted the time left for all the self-supervised experiments. Thus, with respect to the provided hyper-parameters, the model is further tuned on 10% of the data to decrease the training time to something manageable in less than 12 hrs. Our model is then compared to the pre-trained inferenced on the test data. Finally, the only changes that we made to the code provided in [5] were changes to the hyper-parameters in the FastMRI examples U-Net training script and to the evaluation script for inference on the test and challenge set, which do not provide with ground-truth images.

2.2. Self-Supervised hold-out ResNet

Supervised techniques developed for MRI reconstruction are not always practical. A significant reason for this is that it is not always feasible to acquire the fully-sampled datasets required for supervised training. Effective alternatives that do not required fully-sampled reference data are, therefore, highly desirable. We experiment with one such method: a self-supervised hold-out approach with data undersampling. For this approach, data is first masked to simulate under sampling. Then, for each data volume, the set of all *k-space* locations, Ω, is divided into two subsets Θ and Λ [8]. Those in Θ are used in training, and those in Λ are

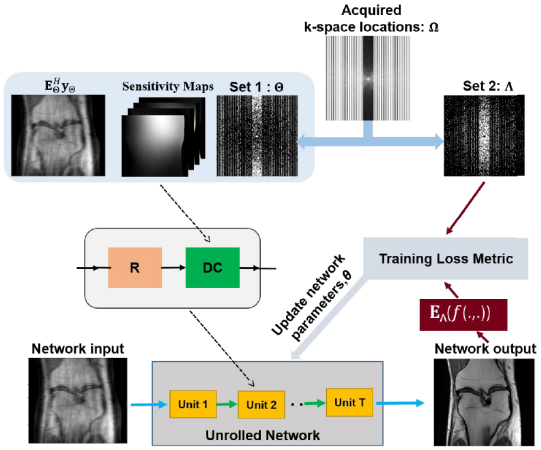


Figure 6. Self-supervised with a hold-out set training process. The Figure and its convention is from [8]. Note that E_Λ is basically our $\mathcal{P}^{Lambda}$

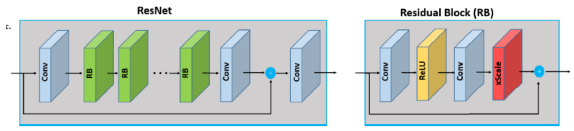


Figure 7. Single-coil ResNet model used in self-supervised with a hold-out set. [8]

used to minimize the following normalized loss:

$$\frac{||\mathbf{u}-\mathbf{v}||_2}{||\mathbf{u}||_2} + \frac{||\mathbf{u}-\mathbf{v}||_1}{||\mathbf{u}||_1} \tag{5}$$

Here, $\mathbf{u}$ corresponds to the *k-space* measurements from Λ, and $\mathbf{v}$ corresponds to the *k-space* from the output of the aforementioned network. The network architecture is an initial convolutional layer, followed by 15 residual blocks (RB) with skip connections. Each RB consists of two separate convolutional layers, with a ReLU activation between them, and a final constant scalar multiplication layer (using value 0.1). All layers throughout had 3x3 kernels and 64 channels. This design is illustrated in Figure 7.

In this case we anticipate problems related to the integration of our implementation in FastMRI's Pytorch Lightning and problems with the enormous time needed for training on the whole dataset. And, indeed integrating the code in FastMRI's Pytorch Lightning framework was problematic, but finally by writing a custom SelfSupervisedModule inherited from the MriModule that uses FastMRI's Pytorch Lightning and implementing the ResNet in a torch.nn.Module, we succeeded in the intent. Much more of a problem was to make sure that the model was

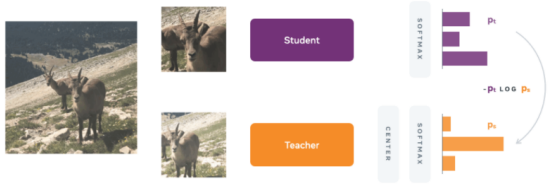


Figure 8. The framework of DINO (Self-supervised learning with Knowledge Distillation). [1]

training and to ensure that it was doing something meaningful, which is a big problem even when prototyping on smaller subsets of the data that takes less than 3 days to train.

2.3. Self-supervised learning with Knowledge Distillation (DINO)

Finally, as additional self-supervised method to experiment with, we decided to explore self-distillation with no labels (DINO) [1], as suggested in the forum. DINO is an astonishing self-supervised knowledge-distillation framework that just came out. It leverages both CNNs backbones like ResNet and Transformers for Vision (ViT) to perform classification tasks on ImageNet (and naturally also segmentation) better than any other self-supervised method thus far.

The framework of DINO, illustrated in Figure 8, is pretty simple, but includes a lot of technical details that are not obvious at all. DINO consists of a student network g_{θ_s} that has to match a teacher network g_{θ_t} output, as in classical knowledge distillation frameworks. Given an input image, both networks output a probability distribution, respectively P_s and P_t, using a Softmax function with a temperature scaling in its exponential to control the distribution sharpness [1]. Given that the teacher is fixed, we learn to match the two distributions by minimizing the cross-entropy loss H between the two over the student parameters θ_s:

$$\min_{\theta_s} H(P_t(x), P_s(x')) \tag{6}$$

To adapt this to the self-supervised framework, DINO constructs different augmented crops of an image using a multi-crop strategy. Specifically, DINO generates for each input image V different augmented crops or views. This set contains two *global* views x_1^g and x_2^g and several *local* views of smaller resolution. The number of views is an hyperparameter to tune, while keeping the *global* views covering large portions of the image (e.g. more than 50%) rescaled to larger resolution (e.g. 224x224) and local views

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Figure 9. The pseudo-algorithm of DINO (Self-supervised learning with Knowledge Distillation). [1]

small portions of the image (e.g. less than 50%) rescaled to smaller resolution (e.g. 96x96). All crops are passed through the student, while just the *global* views are passed to the teacher, enforcing local to global correspondences [1]. So, equation 6 becomes:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')) \tag{7}$$

Finally, while the student is trained by gradient descent on the unlabelled training data, the teacher is frozen and its parameters are constantly update with an exponential moving average, in a momentum approach like that of equation 8. DINO's pseudo-algorithm without multi-cropping is thus shown in Figure 9.

$$\theta_t = \lambda\theta_t + (1-\lambda)\theta_s \tag{8}$$

The problem we had with DINO have to do with the inherent learning tasks. Indeed, DINO is built for classification and not for image reconstruction, which is nothing else than pixel-by-pixel regression. In the beginning, we thought that by adapting the framework to perform an L1-Loss could have been successful. Therefore, we tried implementing the algorithm shown in Figure 9 and represented by equations 7 and 8, by replacing cross-entropy with the L1-Loss and augmenting the input images in a multi-crop approach with 2 *global* views and 2 *local* views. This was done by writing a custom Transform for DINO to apply multi-crops data augmentation to the subsampled input images, re-using the U-Net model of Figure 5 as a backbone, adding a DinoLoss module to compute the DINO loss of equation 7 (but using the L1-Loss instead of cross-entropy) and writing a DinoModule inherited by MriModule to integrate the Pytorch Lightning framework and implement student and teacher training and validation steps, including the momentum of equation 8.

However, after some training, we realized that the multi-crop approach used by DINO to force local-to-global correspondences is unfeasible for image-reconstruction. Indeed, while it makes perfectly sense to try to match student's and teacher's output probability distributions of different crops and augmented views of a dog, because for any view both student and teacher should predict *dog*, it makes no sense to match the reconstructions of different crops of the input subsampled image. Thus, we moved the multi-view approach to the *k-space* using solely *global* views augmented by masking of frequencies. After further training, we realized we had also problems with collapsing of the reconstructions to the zero image. Indeed, DINO contrasts mode collapsing in classification by centering and sharpening continuously the teacher output distribution, which has the effect of balancing the two and produces output distributions that are not too sharp (i.e. collapse to a single class) or too flat (i.e. collapse to the uniform distribution). However, due to the inherent nature of our problem, we don't have a way to prevent collapsing to a uniform distribution of pixels (e.g. a zero image) and that was a major problem that still remains unsolved and that would need further research.

3. Experiments and Results

In order to measure success, we are going to use in our experiments the same evaluation metrics that are currently being used in the literature, given a reconstructed slice $\hat{v}$ and the ground-truth v, with means μ_v and $\mu_{\hat{v}}$, variances σ_v^2 and $\sigma_{\hat{v}}^2$ and covariance $\sigma_{\hat{v}v}$:

- *Normalized Mean Square Error* (NMSE):

$$NMSE(\hat{v}, v) = \frac{||\hat{v} - v||_2^2}{||v||_2^2} \tag{9}$$

- *Peak Signal-to-Noise Ratio* (PSNR)

$$PSNR(\hat{v}, v) = 10 \log_{10} \frac{max(v)^2}{MSE(\hat{v}, v)} \tag{10}$$

- *Structural Similarity* (SSIM)

$$SSIM(\hat{v}, v) = \frac{(2\mu_{\hat{v}}\mu_v + c_1)(2\sigma_{\hat{v}v} + c_2)}{(\mu_{\hat{v}}^2 + \mu_v^2 + c_1)(\sigma_{\hat{v}}^2 + \sigma_v^2 + c_2)} \tag{11}$$

Although, NMSE is the main evaluation metrics in the current literature, it tends to favor much more smoothness rather than sharpness, which in MRI may be problematic due to the loss of fundamental details for diagnosis. Thus, we also report PSNR, which gives an idea of the ratio

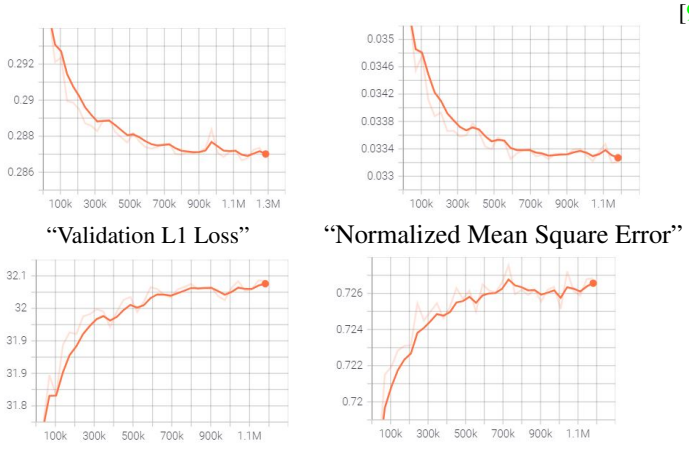
of powers between the maximum image intensity and the noise, and SSIM, which measures similarity between the images' structures [9]. Nonetheless, these metrics do not necessarily ensure the level of details in the MRI reconstruction required for proper diagnosis, which would need sharp and detailed images. Therefore, we will be training using L1-loss (which is also faster to compute) to encourage pixel-to-pixel identity between reconstructed images and ground-truth:

$$L_1(\hat{v}, v) = ||\hat{v} - v||_1 \tag{12}$$

3.1. Re-training from scratch the U-Net model

In our implementation the main intention is for the model to generate good quality MRI image with fewer samples *k-space* image slices by FastMRI scan . In order to simulate the data for poorly sampled input *k-space* image data we downsample the fully/sampled input kspace image by performing a sub-sampling mask as shown in Figure 4. After passing the image through the U-Net layer, the final output is compared against the ground-truth image, which is the image created by inverse DFT from the fully-sampled *k-space* slice, as shown in 1. The loss is calculated using the L1 Loss as shown in equation 12. The gradient based optimizer RMSRrop is used for training the network and updating the weights of the network.

We ran the base experiments by training the model on the single-coil knee dataset [4]. The training is performed with an acceleration factor of 4, with random masking applied to the *k-space* input image to simulate the undersampled input. The acceleration determines the rate of undersampling: higher the value, greater is the amount of undersampling. The complete training on 32 core CPU and 64 GB RAM took almost 3 days to complete. The model is trained in 50 epochs using the training and validation sets, as shown in the following graphs:



"Validation L1 Loss" "Normalized Mean Square Error"

"Point to Noise Radio" "Structural Similarity Index"

Figure 10. Training Metrics. [9]

From Figure 11, we can see that the loss decreases and then starts saturating after 900K steps. Similarly we see that NMSE error also follows the same pattern and becomes stable after 1000K steps. The PSNR and the SSIM also keeps improving in the training and becomes saturated 1000K steps. As shown in Figure 11, we see significant improvement with respect to noise reduction. The main challenge faced is the long training time it takes for the model to learn. In the result generated using Tensorboard, we see that training metrics improves over iterations in the validation set as shown in Figure 10.

"Reconstructed Image" "Target Image" "Error Image "

Figure 11. Reconstructed and ground-truth images from Validation set. [9]

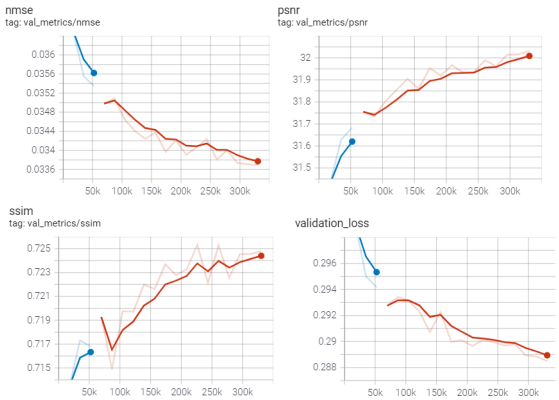


Figure 12. Training Metrics with 20 epoch and 0.1 Sampling Rate. [9]

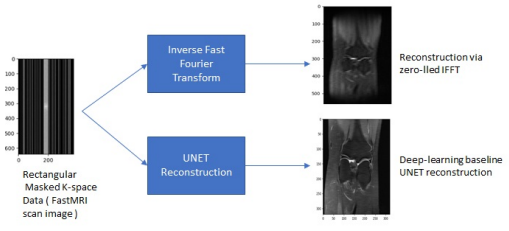


Figure 13. IFFT and Unet Generated Image. [9]

Also the testing and challenge images did not have the ground-truth images, due to which we could not compute the metric of performance from the testing. We again

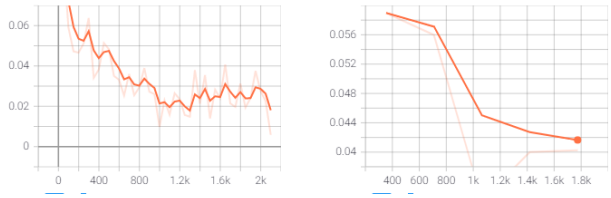
trained the model with 0.5 sampling rate and with 20 epoch, which took hours to complete. The training did not perform well on the validation set due to limited Sample and very less epoch. Please the training metrics at 12. We also observed that the loss and the improvement does stabilize with within the limited epoch count of 20. Also the quality of the image generated from same Validation date in Figure 11 is not having good quality.

3.2. Self-Supervised hold-out ResNet

Experimental results with the self-supervised method were disappointing and ultimately unsuccessful. Code was implemented that successfully run through a training routine. It executes the entire forward and backward training routine, and is able to save model state dictionaries for each epoch. However, it must also contains subtle errors that there was not time to fully resolve, as NaNs are returned during validation. A significant contributing factor was the large dataset size combined with the lack of available resources to effectively train and develop training routines. Team members with access to GPUs were assign to other project tasks and unable to allocate time to run the training code after it was developed. Additionally, while plenty of credits to Google Cloud Compute resources were available, 17 attempts were made over the last week of the project to acquire a GPU compute instance for training and testing, over multiple zones advertised to have GPU instances accessible. All of these failed, with the error message asserting resources were not available. This does not include additional issues that were the result of quota configuration. The eventual result was that code testing, model training, and model testing all had to be done on limited hardware. Even with a highly restricted dataset, the took and enormous amount of time, require that assumptions be made about the working state of the training routine (i.e., execution of the loop of batches through forward and backward calls works, so training must be working). An important lesson to be taken away from this is a thorough assessment of data implications and compute resource availability at the initial stage of the project. These were only slightly considered when initially putting together the project plan. In particular, resource availability with strong guarantees on at least the minimal allocations was sorely missed thorough this project. A takeaway is that additional emphasis should be applied on analyzing project timeline feasibility given data requirements in cases when such resource availability guarantees are not available.

3.3. Self-supervised learning with Knowledge Distillation (DINO)

As already explained in Section 2.3, unfortunately, despite many attempts, we were not able to make DINO train effectively. The reason why we think so is that the reconstructed image was close to the zero image, so a trivial solution due to collapsing of the teacher-student networks to a uniform distribution of the pixels. Figure 14 clearly shows the failing of the training trough the training and validation losses, which quickly decrease (in just a few iteration during the first epoch) to a small value. That is significant to understand that the main problem of our approach is the loss and its collapsing immediately to a trivial solution in absence of any supervision. If time had allowed, we would have experimented with different losses (maybe, by thinking each pixel as a class per-se we could have used cross-entropy to try to stay close to the main concepts presented in [1]) or could have found other self-supervised methods that were more targeted to this kind of application or even trained in a semi-supervised approach, by just providing a small part of the ground-truth reconstructions. However, the problem of applying self-supervised learning to MRI image reconstruction remains fundamentally hard. Especially when not having enough resources to tackle the mole of data.



"Training Loss " "Validation Loss "

Figure 14. Example of failure in training DINO, probably due to collapse of the output to a trivial solution [9]

4. Discussions

In concluding, we truly regret not to have been able to achieve satisfying results in the self-supervised approaches. This was due to several problems, whose first was the non-collaboration of some of the contributors, which has caused a lot of delays in the implementation and the actual experiments, as well as it has subtracted a lot of time to the ones actually contributing. Second, our resources were not enough with respect to the time available and to the mole of the data. Nonetheless, we are satisfied by our results in the supervised approach, where we can reproduce good-quality reconstructions. Despite the problem is fundamentally hard, we are sure that if we had concentrated our efforts on just one SSL approach, we would have been able to successfully train a model with no supervision at all. However, we have found in the problem, great interest and utility for the humanity overall and we think that exploring in the future SSL approaches to MRI reconstruction, like the ones we tried to explore, could lead to enormous benefits for making MRI cheaper, more accessible and of higher quality for diagnosis of dangerous diseases.

Student Name	Contributed Aspects	Details
Enrico Zerilli	Project Setup, Study of the Material, Self-Supervised with Knowledge Distillation (DINO)	GitHub Setup, Study of the Material, Self-Supervised with Knowledge Distillation implementation, training and experiments. Report: everything but Sections 3.1, and 3.2.
Sudipto Lodh	Re-train from scratch the supervised U-Net model	Re-train from scratch the supervised U-Net model and corresponding experiments. Support in training DINO. Report: Section 3.1 only.
Robert Bartel	Self-Supervised Hold-out ResNet	Self-Supervised Hold-out ResNet implementation. Attempts to acquire external compute resources for training and testing. Unsuccessful training attempts (limited hardware). Report: 2.2 and 3.2. Some of the Work Division section also.

Table 2. Contributions of team members.

5. Work Division

In Table 2 you can see the contributions of each of the members. While that summarized things, it should be again pointed out that the bulk of the report writing and team organization was performed by Enrico Zerilli. Team communication overall was difficult, due to a combination of circumstances (individual circumstances outside the course, time zone differences, etc.). Also, we found out that having just one GPU with 64 GB of RAM was not enough to perform all the experiments that we had to do and coordination of training experiments was often a problem in this regard. Nonetheless, we managed to complete a satisfying project.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 3, 5, 6, 8

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[3] Florian Knoll, Jure Zbontar, Anuroop Sriram, Matthew J. Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, James Pinkerton, Duo Wang, Nafissa Yakubova, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastmri: A publicly available raw k-space and DICOM dataset of knee images for accelerated MR image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1):e190007, 2020. 2

[4] NYU School of Medicine and Facebook AI Research (FAIR). The fastMRI dataset. https://fastmri.med.nyu.edu/. Accessed: 2021-04-25. 2, 3, 7

[5] NYU School of Medicine and Facebook AI Research (FAIR). The fastMRI repository. https://github.com/facebookresearch/fastMRI. Accessed: 2021-04-25. 3, 4

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 3

[7] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig. ESPIRiT–an eigenvalue approach to autocalibrating parallel mri: where sense meets grappa. *Magnetic Resonance in Medicine*, page 990–1001, 2014. 2

[8] Burhaneddin Yaman, Seyed Amir Hossein Hosseini, Steen Moeller, Jutta Ellermann, Kâmil Uğurbil, and Mehmet Akçakaya. Self-supervised learning of physics-guided reconstruction neural networks without fully sampled reference data. *Magnetic Resonance in Medicine*, 84(6):3172–3191, Jul 2020. 3, 4, 5

[9] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J. Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, Marc Parente, Krzysztof J. Geras, Joe Katsnelson, Hersh Chandarana, Zizhao Zhang, Michal Drozdzal, Adriana Romero, Michael Rabbat, Pascal Vincent, Nafissa Yakubova, James Pinkerton, Duo Wang, Erich Owens, C. Lawrence Zitnick, Michael P. Recht, Daniel K. Sodickson, and Yvonne W. Lui. fastMRI: An open dataset and benchmarks for accelerated MRI. 2018. 1, 2, 3, 4, 7, 8