FastMRI - Project Report : CS 7643 O1 - Fall 2020

Dhruv Kapoor
Georgia Institute of Technology
dkapoor7@gatech.edu

Jeremy Martinez
Georgia Institute of Technology
jmartinez91@gatech.edu

Venkateswaran Tarakad
Georgia Institute of Technology
vtarakad3@gatech.edu

Vivek Iyer
Georgia Institute of Technology
viyer@gatech.edu

Abstract

This paper attempts to replicate the results for the single-coil knee scan in [5] using the U-Net architecture and making further adjustments to the underlying model. After evaluating the results of U-Net, we attempted to come up with competing models by replacing the underlying network architecture with several other models instead, and comparing the losses across all models. The models explored include NNRET U-Net, R2U-Net, ResNet, Dense U-Net, Residual Dense U-Net, Attention U-Net, U-Net++, and U-Net++ with deep supervision. In addition to testing other models, augmenting the learning by using other methods (like the k-space correction proposed in [1]) were also explored. We chose this project because improving on MRI technology can have tremendous real world impact and the unsupervised aspect of it was appealing to us. Single-coil knee images were chosen over multi-coil in order to save on training time, which allowed us to trial a large number of models.

1. Introduction/Background/Motivation

FastMRI is an open source project focused on accelerating magnetic resonance imaging (MRI). It attempts to do so by acquiring fewer measurements, which would lead to faster imaging, lower medical costs, minimize stress of patients during the procedure, and open the possibility for MRI scans to be applied during situations where time is a constraint. Our team forked this project and applied a handful of models in an attempt to improve accuracy. We trained our models on the knee single-coil images. By doing so, we gained insight into the U-Net model structure and how variations of it can affect image segmentation. Facebook, in collaboration with NYU Langone Health, maintains the original project and also maintain a leaderboard for models that achieve the lowest loss. Although we did not beat the current record, we addressed a problem that provided valuable takeaways and expanded the breadth of models that can be used to compete with U-Net.

Today, these models are trained on high-powered machines, across multiple GPUs. This is required because the training time can take several hours to complete. Often times, we would train our models overnight. Due to the time-consuming training periods, we were limited in how many permutations of hyper-parameters we were able to tune. To meet hardware requirements, some team members were able to provide access to their workstations rather than having our training run on cloud resources.

At a high level, we are applying a convolutional neural network (CNN) to deconstruct and then reconstruct images. CNNs are traditionally used for classification. However, in biomedical image processing, we are in fact altering this to run classification on each individual pixel. This is extrapolated across the entire image to accomplish image segmentation.

If we can successfully rebuild useful MRI images from undersampled data, then we can have a profound impact in the medical field from a financial as well as a diagnostic standpoint. As stated before, this could reduce cost, reduce patient stress, and provide versatility for situations where MRI scans can take place.

NYU Langone Health's dataset was used during development [7]. This data is provided for free, upon request. It is in the form of k-space data corresponding to MRI imaging slices. NYU provides imaging data for both knee and brain scans. Knee data is available in single and multi-coil, whereas brain data is available only for multi-coil. The size of the multi-coil data was much larger. The amount of computational resources required to train models on that data was a deciding factor in moving forward only with knee single-coil. The single-coil knee, multi-coil knee, and multi-coil brain training data zipped files were 88GB, 931GB and 1,228GB, respectively.

2. Approach

The specific models that we implemented for fastMRI, which we enumerate in greater detail in section 3, include NNRET U-Net, R2U-Net, ResNet, U-Net++, U-Net++ with deep supervision, Dense U-Net, Residual Dense U-Net, and Attention U-Net. We extended functionality from the base model (U-Net) in order to both test as well as train each model. Code for each of these tasks, across all models, can be found in the experimental directory of the codebase while the addition of new models and altering existing models (including hyperparameter tuning) mostly took place in the models directory [13]. We envisioned many of these models performing well, and at least one of them competing with U-Net. Our approach was new in that it introduced several models into the fastMRI codebase that previously did not exist. By leveraging our increasing familiarity with fastMRI, plugging-and-playing with different models became smoother. With this project, we've seen how valuable it can be to have a modular API with a dynamic model where you can benchmark certain network structures against others.

Given that we are experimenting with so many different models, we needed to consider allocating enough time to train all of these models individually. We anticipated training to be a problem, given the amount of time it requires to train an individual model. Now extrapolate this across eight newly introduced models and we potentially have an issue. Luckily, we had access to two separate machines with multiple GPUs so that we could constantly be running overnight builds. However, we did encounter problems with these overnight builds crashing. When this would happen, we essentially would lose an entire evening's worth of work. We also ran into possible issues with the upstream codebase when training with 3+ GPUs on a single machine. We found that the training loss was printing incorrectly and inconsistently, so we needed to throttle our own throughput to make up for it.

Something we did not anticipate early in the planning phase was the possibility of internal bugs in the fastMRI codebase. We had issues training U-Net originally across Mac OS X and Windows PCs due to an error that ended up being related to internal randomization. The issue was opened up by us in a Github issue [8] and the fix was also submitted by us in a pull request [9] before it was approved and merged into the master branch of the upstream repository. Another issue was that of the GPU bug having incorrect output when the number of GPUs used is 3+. The solution to this issue is less clear, and seems to be a known issue, minor enough to not necessitate a fix. We submitted an issue for this as well in the Github repository [3].

Any changes to the source code for the Facebook repository as well as our codebase are in the upstream Facebook Github repository [6] and our forked repository [13] respectively. The README.md file in our codebase details the steps required to run the experiments in the following section.

3. Experiments and Results

The main indicator of success for any given model was a low training and validation L1 loss. We ran the same experiment (training on knee single-coil) on every model to make this comparison easier. The final results can be found in table 1 with the training and validation trends recorded in Figure 1 and Figure 2 respectively. As you can see here, we achieve parity with U-Net. Some of our models, like U-Net++ and U-Net++ with deep supervision had lower training losses, but had issues translating that into an acceptable validation loss, suggesting overfitting. For more detailed information on each model and how their structure supported/discouraged learning, see the corresponding subsections below.

3.1. NNRET U-Net

Nearest-Neighbor Re-sampling Based Elastic-Transformed (NNRET) U-Net is a U-Net variant that swaps out one key block in the original model [2]. Rather than a transposed convolutional block used for upsampling, a KNN approach is used with a scaling factor of 2. For a consistent comparison, the parameters used for the original U-Net implementation were untouched and only these blocks were changed. The original authors of NNRET U-Net found a large amount of computational savings using this approach and were able to surpass the performance of U-Net on MRI data. Unfortunately, we did not see the same outcome and the difference was deemed large enough to forgo further optimization of NNRET U-Net. Although it had the sharpest decline in loss in both Figure 1 and Figure 2 when compared to any other algorithm, the result looked to plateau at local minima. Why NNRET U-Net failed to compete with U-Net is likely because the transposed convolution that had important pixel information that was written over when using KNN instead and also because the learning rate may have been too low, causing the plateau of loss. The result was a much faster algorithm, but one that was more crude.

3.2. R2U-Net

Recurrent Residual Convolutional Neural Network (R2U-Net) was first developed in 2018 as an alternative to U-Net in image segmentation [10]. Originally, R2U-Net was tested on three benchmark datasets - blood vessel segmentation in retina images, skin cancer segmentation, and lung lesion segmentation. The original authors of this model claim that "feature accumulation with recurrent residual convolutional layers ensures better feature representation for segmentation tasks." For this reason,

	# Params	Iter Size	Param Size	Total Size	Train Loss	Val Loss
U-Net	7,756,097	1046.09	29.59	1076.07	0.127	0.289
NNRET U-Net	7,059,777	952.34	26.93	979.67	0.150	0.316
R2U-Net	11,682,113	1427.34	44.56	1472.30	0.126	0.289
ResNet	427,584	14.84	1.63	16.87	0.191	0.319
U-Net++	9,162,753	1729.69	34.95	1765.03	0.120	0.361
U-Net++ Deep	9,162,753	1729.69	34.95	1765.03	0.120	0.353
Dense U-Net	363,413	847.68	1.39	849.45	0.143	0.291
Residual Dense U-Net	205,134	1138.30	0.78	1139.48	0.142	0.291
Attn U-Net	7,844,829	1213.27	29.93	1243.59	0.124	0.289

Table 1. Model comparison. The size columns are in MB.

we thought it would be interesting to see how it fared in fastMRI.

R2U-Net introduces recurrent convolutional layers to the U-Net architecture, modifying it slightly. A good comparison of these two architectures side-by-side is visualized in figure 4 of [10]. Our intuition was correct and this algorithm was able to generate similar results when compared with U-Net.

3.3. ResNet

The model summary for our ResNet implementation can be found in our repository. It consists of a reflection padding layer which pads the outer edges with its own reflection. It has advantages over zero padding helping avoid edge effects. The image is then passed through Conv2d and ReLU. This sequence is repeated 6 times (No ReLU in the last layer). U-Net in comparison is a bigger model consisting of 98 layers versus just 17 for ResNet. Key comparisons are outlined in table 1. This algorithm was unable to sustain the loss metrics of U-Net, possibly due to edge effects not being a major concern for this scenario. The Resnet model implementation was adapted from [12].

3.4. U-Net++

UNet++ consists of an encoder and decoder that are connected through a series of nested dense convolutional blocks [14]. The main idea behind UNet++ is to bridge the semantic gap between the feature maps of the encoder and decoder prior to fusion. What distinguishes UNet++ from U-Net is the re-designed skip pathways (shown in green and blue in figure 1 of [14] that connect the two sub-networks and the use of deep supervision (shown in the same figure in red).

UNet++ was run on the dataset both with and without deep supervision. Though the model trained very well and achieved the least training loss among all other models, the validation loss did not converge, indicating that the model was over-fitting the data. U-Net++ is larger in comparison to U-Net (Table 1), and takes much longer to train. It is possible that hyper-parameter tuning might lead to better results with U-Net++, however given the large training time we were unable to attempt this. The best validation loss for U-Net++ was 0.35764 vs 0.28857 for U-Net, while the best training loss was 0.120 compared to U-Net's 0.127.

3.5. Dense U-Net

Dense U-Net is an adaptation of U-Net model and it has 118 layers. The model summary for our Dense U-Net implementation can be found in our repository. The rest of the layers are a combination of the layers from the table. The Dense U-Net model was adapted from [4]. Key difference between U-Net and Dense U-Net is shown in table 1. Even though Dense U-Net is denser than original U-Net, the number of parameters for Dense U-Net was reduced for faster training. Increased parameters for Dense U-Net did not show any significant improvement in Validation loss. The best validation loss for Dense U-Net was 0.29145 vs 0.28857 for U-Net.

3.6. Residual Dense U-Net

Residual Dense U-Net is a variation of Dense U-Net model and has 188 layers. The model summary of Residual Dense U-Net is similar to Dense U-Net and is a combination of layers from table **??**. A comparison of Residual Dense U-Net, Dense U-Net and U-Net is outlined in table 1. The parameter size is the number of parameters per layer which is only 0.78 for Res Dense U-Net. The total number of parameters for the model was kept low for faster training. Again this did not impact the validation loss. Residual Dense U-Net model was also adapted from [4]. The best validation loss for Residual Dense U-Net was 0.29118 which was comparable to Dense U-Net.

3.7. Attention U-Net

Attention U-Net builds an attention model on top the standard U-Net architecture [11]. Attention gates are incorporated into the standard U-Net architecture to highlight features that are passed through the skip connections. This is performed right before the concatenation operation to

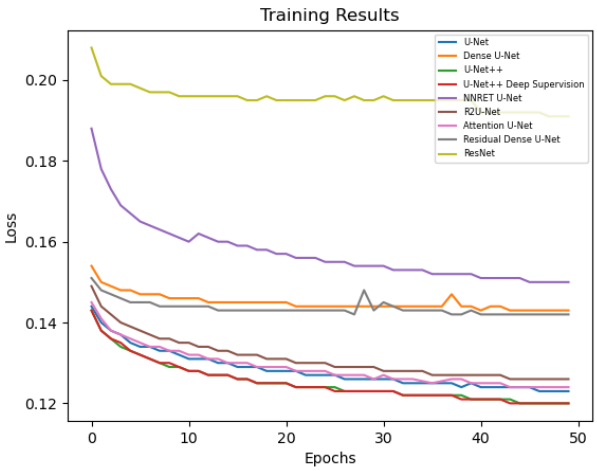

Figure 1. Training Loss

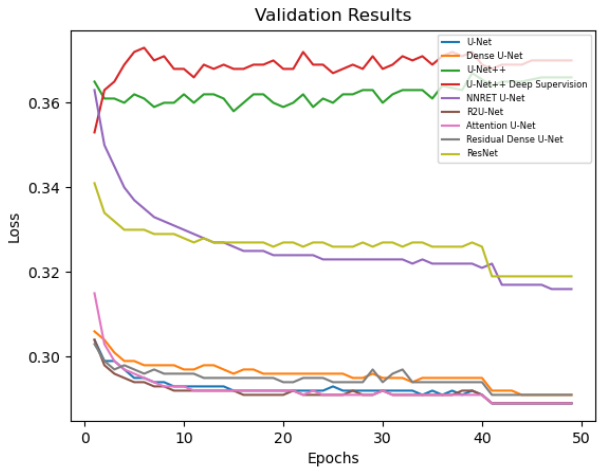

Figure 2. Validation Loss

merge only relevant activations as seen in figure 1 and 2 of [11]. As seen in the learning plots 1 and 2, the model trains as well as U-Net and the beats U-Net in validation performance during the first few epochs. The best validation loss for Attention U-Net was 0.28883 vs 0.28857 for U-Net.

3.8. Image reconstruction

Image reconstruction was done on the validation data set because the test set did not have a target image to compare to. Since the validation set was not used for training, this was acceptable. The best saved model was used for reconstructing the image. Each validation data file had 36 slices of fully sampled kspace data for a patient. The fully sampled kspace data is converted to an undersampled data by randomly applying a mask. During the test, this undersampled kspace data is converted to an image and passed to the model. This undersampled input image and the model output image is saved along with the target image. We used slice number 22 to reconstruct because this slice resembled a complete knee. The input image and the target image is same for all the models. The output image for all the models are shown in figure 3

4. Work Division

See table 2.

5. Conclusion

Results for all models can be found in table 1. In the end, we succeeded in replacing U-Net with competing models. Namely, R2U-Net and Attention U-Net were able to achieve L1 losses comparable to U-Net. Looking at Table 1, we can see that these two models are larger, and although Attention U-Net is only slightly so, R2U-Net is the largest model we trialled. Cross referencing the two tables, the size and L1 losses seem to have no strong correlation, leading us to believe that model complexity is not a defining characteristic of performance for this reconstruction problem. While it is expected that further tuning Attention U-Net or R2U-Net would exceed the performance of U-Net, tuning the former is more favoured due to its much smaller training time. After implementing these models, we were able to discover how unique vanilla U-Net's results were and we discovered that all the models we trialled, aside from the U-Net++ variants, had similar training and validation trend lines, with a plateau at the end of epoch 50. This may be indicative that a more extreme change is needed to the underlying model.

6. Further research

Given more time, we would like to take the research in the following four directions. First, we would have liked to have access to even more high performance stations with powerful GPUs. This way, we could have tested more permutations of hyperparameters and we wouldn't need to shy away from even larger models. Second, we would like to apply k-space correction, similarly to that of [1] as a form of augmentation. This would sit on top of our trialled models to see if the results are improved and, if so, if it's for all of our models. Thirdly, we would like to take a trained model and apply it in a classification setting in an attempt classify these reconstructed images under a binary classification of certain injuries, such as MCL/ACL tears. This would give another metric of evaluation rather than only the L1 loss. Finally, we would like to train on VarNet, especially since it uses SSIM, which is expected to circumvent the issues we encountered while trying to train with 3+ GPUs.

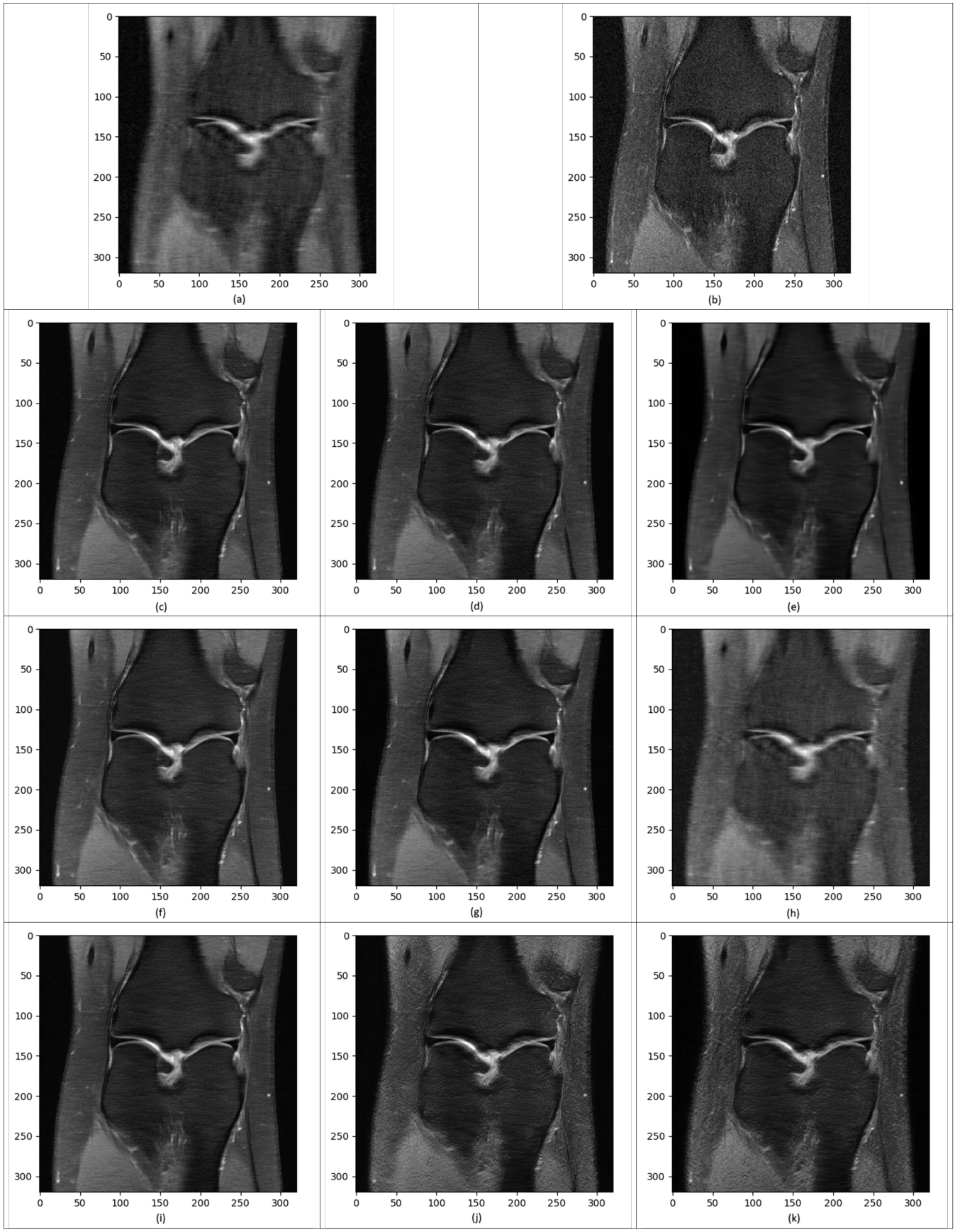

Figure 3. Reconstructed images (a) Input (b) Target (c) Attention U-Net (d) Dense U-Net (e) NNRET U-Net (f) R2U-Net (g) Residual Dense U-Net (h) ResNet (i) U-Net (j) U-Net++ (k) U-Net++ with deep supervision

Student Name	Contributed Aspects	Details
Dhruv Kapoor	- Found/Fixed an issue for the original fastMRI codebase for running U-Net [8], [9] - Implemented NNRET U-Net from [2] - Created the script to create figure 1 and 2 - Filled in missing sections and edited the report and README	Issues were incurred with the training for U-Net in the original repository. The issue was resolved after the changes were merged. From there, was the first member to implement a new model, NNRET U-Net. Worked on extracting graphs afterwards. Spent time filling in the report/README with other members.
Jeremy Martinez	- R2U-net - testing - final report - writing final report	Researched and implemented R2U-net. Tested training on remote GPU server. Wrote tests for R2U-net. Implemented significant portion of final report.
Venkat Tarakad	- resnet [12], Dense U-Net and Residual Dense U-Net [4]. - Run the model on 2080 ti. - Image reconstruction. - report and README.	Researched and implemented 3 models Resnet, Dense U-Net and Residual Dense U-Net. Helped run Resnet, Dense U-Net, Residual Dense U-Net and NNRET on a machine with GPU and posted the results on Github. Implemented image reconstruction on the trained models for all the models.
Vivek Iyer	- U-Net++ [14] and Attn U-Net [11] - Hyper-parameters tuning - Model training - Report and GitHub README	Researched and implemented UNet++ and Attention UNet. Helped train models on my machines with multiple GPUs, and tuned hyper-parameters for these models. Updated sections in the report associated with the models I worked on, and worked with the team to ensure the accuracy and quality of report and the README

Table 2. Team Member Contributions

References

[1] Sung Min Lee Sungchul Lee Chang Min Hyun, Hwa Pyung Kim and Jin Keun Seo. Deep learning for undersampled MRI reconstruction. *Physics in Medicine and Biology*, 63(13), 2018. 1, 4

[2] S M Kamrul Hasan and Cristian A Linte. A Modified U-Net Convolutional Network Featuring a Nearest-neighbor Re-sampling-based Elastic-Transformation for Brain Tissue Characterization and Segmentation. *Proceedings. IEEE Western New York Image and Signal Processing Workshop*, Oct 2018. 2, 6

[3] Vivek Iyer. Incorrect training loss when running across multiple GPUs (with ddp), 2020. 2

[4] H. Chung J. Lee, H. Kim and J. C. Ye. Deep Learning Fast MRI Using Channel Attention in Magnitude Domain. pages 917–920, 2020. 3, 6

[5] 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 Jure Zbontar, Florian Knoll and Yvonne W. Lui. fastMRI: An Open Dataset and Benchmarks for Accelerated MRI. *ArXiv e-prints*, 2018. 1

[6] 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 Jure Zbontar, Florian Knoll and Yvonne W. Lui. fastMRI Codebase, 2018. 2

[7] 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 Jure Zbontar, Florian Knoll and Yvonne W. Lui. NYU fastMRI Dataset, 2020. 1

[8] Dhruv Kapoor. Failing unit tests due to pickle command., 2020. 2, 6

[9] Dhruv Kapoor. Fix to the issues of running unit tests., 2020. 2, 6

[10] Chris Yakopcic Tarek M. Taha Vijayan K. Asari Md Zahangir Alom, Mahmudul Hasan. Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation. *ArXiv e-prints*, 2018. 2, 3

[11] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018. 3, 4, 6

[12] Jackson L Sami K. Improving Undersampled MRI with Deep Learning. 3, 6

[13] Jeremy Martinez Vivek Iyer, Dhruv Kapoor and Venkateswaran Tarakad. MRI Puns Codebase, 2020. 2

[14] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. pages 3–11, 2018. 3, 6