FastMRI Project Report : CS 7643 (Spring 2021)

Mohit Chawla
Georgia Institute of Technology
mohitchawla@gatech.edu

Prashant Sharma
Georgia Institute of Technology
prashant.sharma@gatech.edu

Vaneesh Bahl
Georgia Institute of Technology
vbahl8@gatech.edu

Abstract

This paper describes the attempt by our project team to match or exceed the accuracy of the baseline U-Net model on single-coil MRI knee scans. Once we established the accuracy of the baseline model, we tried several challenger models which substitute or enhance the core network architecture of U-Net using other deep learning mechanisms – models like R2U-Net, Dense U-Net, Attention U-Net, and Nested U-net. We also used ResNet to explore how a state-of-the-art model that works well with image classification would work with a biomedical image segmentation task. Beyond these, techniques like KNN which can be used in conjunction with U-Net and can potentially deliver better performance were studied. We opted for the FastMRI project because accelerating MRI can produce incredible real-life impacts. Speeding up Magnetic Resonance Imaging (MRI) using fewer measurements can reduce imaging costs, reduce patient discomfort and stress and enable use in situations where scans with current technology are not feasible. Due to limited compute resources (GPUs), we tried to keep the training times in check by selecting the Single-coil knee images over multi-coil images. Complete code and other elements are available at our Github repository

1. Introduction/Background/Motivation

Magnetic Resonance Imaging (MRI) is a useful diagnostic test for a large number of medical conditions. However, the process of acquiring an MRI scan can take 30 minutes or more which can lead to patient discomfort and stress, artifacts from patient motion, low throughput, and high exam costs [7]. Consequently, any approaches which can speed up the process would be a valuable contribution to the medical field.

MRI scans do not use direct imaging but rely on applying magnetic fields and taking multiple measurements of the body's electromagnetic response fields. These measurements must then be recombined in some way to generate an image. Approaches that use Computer Science rely on acquiring fewer measurements to speed up the process. However, this introduces aliasing artifacts which must be eliminated during image reconstruction to get a diagnostic quality image. In general, Machine Learning approaches use prior knowledge to address the problem of aliasing during reconstruction [7].

The fastMRI project is a collaboration between Facebook and NYU School of Medicine to encourage ML research into speeding up MRIs. It provides multiple datasets and a code-base with data-manipulation utilities and a baseline model to benchmark the results. Our team's objective was to build and run new models using this code base to improve accuracy for image reconstruction. We ran these models on the emulated single-coil knee scans. While we were unable to achieve a significant improvement in accuracy due to time and computational resource constraints, we developed a better insight into concepts covered in this course – concepts like fully-convolutional networks, recurrent networks, attention, residual connections etc.

We faced several challenges during the implementation of this project which forced us to alter/limit the scope of our work in several important ways. For more details, please refer to section 4 of this report.

Successful image reconstruction of under sampled images would reduce the cost of MRI and potentially make it available to a larger number of people. It would also reduce the time taken for each exam which would result in less stress and discomfort for the patients and a higher throughput for centers that administer MRI exams. Finally, it could be a lifesaver in situations that require a quick turnaround – during natural and man-made disasters, for instance.

The MRI dataset is provided free of charge on request

by NYU Langone health. MRI measurements consist of k-space data corresponding to a sequence of samples called slices. There are three data sets available – a multi-coil brain MRI dataset, a knee multi-coil dataset and an emulated knee single-coil dataset. Multi-coil datasets being around 1 TB of data each would require significantly more computational resources than the single-coil data set. So, the single-coil knee MRI data was used in our experiments to keep the computational overhead low. The training, validation and test sets in this data were 88 GB, 17 GB and 7 GB respectively.

2. Approach

Models used for medical image processing typically use Fully Convolutional Networks (FCNs) for semantic segmentation. The encoder layers are used to extract and interpret the context and the decoder layers allow for localization. Skip connections are used to recover fine grained spatial information lost in the encoder layers [3]. FastMRI's baseline model U-Net is one such architecture. In addition to being an FCN, it uses excessive data augmentation to make the network invariant to deformations and reduce the need for training examples [5].

Our approach was to evaluate variants of the U-Net architecture by extending the code provided in the fastMRI project. Several models like Attention U-Net, R2U-Net, Dense U-Net and Netsed Unet were tried. In addition, we tried ResNet to see how it would work on a segmentation task. Details of these models are provided in the next section. We also explored the ResU-Net++ model but ran out of time to train more models.

We anticipated at least one of these models outperforming the baseline model since each used some kind of enhancement (attention gates, recurrent nodes etc.) that could potentially improve its accuracy. Also, since these models are neither included in the fastMRI code nor show up in the top entries of the leader board, each model constitutes a new approach to our knowledge [7].

We anticipated two kinds of problems. First, we anticipated requiring a lot of time to train models. Second, we anticipated some of these models to over or under fit or for their loss to not converge at all. The training time issue hit us as soon as we ran the baseline model. It took 50+ hours to train for the default 50 epochs. We ran studies of all subsequent models and discovered some of them would take several hours per epoch. We decided to scale the training down to 20 epochs each. However, even this proved to be a problem since we were using Google Cloud VMs to train the models and they kept being randomly pre-empted often resulting in lost weights and checkpoints. A full description of our experience is provided in later sections. We also discuss our experience with under and over-fitting and loss convergence in the experiments section.

One thing that we had not anticipated were issues with reconstructing the image. After some hit and trial, we decided to reconstruct only slice 22 since it shows a complete knee. But as of this writing, we were still trying to figure out why the reconstructed images were of poor quality – some to the point of being washed out.

3. Experiments and Results

We trained and tested 6 models – U-Net, Attention U-Net, Recurrent Residual U-Net (R2U-Net), Dense U-Net, U-Net++ and ResNet on the single-coil knee data and compared their performance. The evaluation metric for all the experiments we did was the L1 validation loss – the lower the loss, the better the model. All models were trained for 20 epochs due to computational constraints. We believe 50 epochs would have produced a better comparison since the training and validation loss was slowly trending down, but the trend is clear even with 20 epochs. The outcome is displayed in Figure 1 with the graphs for training and validation performances tracked in Figure 2. We also compared the number and size of trainable parameters for each model (see Figure 1). We wanted to attempt a deeper analysis but were hamstrung due to several factors which we discuss in the Challenges section. We observed that while a few models came close to the performance of the U-Net model, none exceeded it. It seems that the added elements in these models failed to improve learning, and hence, accuracy with only 20 epochs of training. Given a longer training period, some of them might deliver superior performance. For further information on each model and how their architecture was derived from Unet or modified by our team in our quest to achieve state of the art results, please refer to the corresponding subsections below.

3.1. U-Net

U-net is a fully convolutional neural network architecture designed primarily for image segmentation. It has some very desirable features – it can create highly detailed segmentation maps using very limited training samples and trains much faster than other image segmentation networks. Since availability of labeled training data is often an issue in medical imaging, this architecture has become very popular in this field.

The architecture uses self-supervised learning with strong data augmentation for image segmentation. It consists of a contracting encoder path and an expansive decoder path that yields a symmetric, U-shaped architecture. 3 x 3 convolution, ReLU and 2 x 2 maxpool layers with a stride of 2 are used in the encoder while upconvolution layers replace maxpool in the decoder. In the expansive path, cropped feature maps from the contracting path transferred via skip connections are concatenated with upsampled features in the expanding path followed by two convolution

	Trainable Parameters	Estimated model params size	Training Loss	Validation Loss
Unet	7.8 million	31.02 Mb	0.23	0.287
Dense Unet	363k	1.45 Mb	0.236	0.292
Attention	7.8 million	31.8 Mb	0.262	0.295
Nested Unet	9.2 million	36.65 Mb	0.228	0.333
R2Unet	7.8 million	31.02 Mb	0.218	0.288
Resnet	427k	1.7 Mb	0.271	0.321

Figure 1. Model Comparisons

layers with ReLU. Finally, a 1 x 1 convolutional layer is used to map each feature vector into the desired number of classes [5] [6].Unet had the best validation loss of 0.287 after 20 Epochs.

3.2. R2Unet

R2U-Net is a combination of Residual U-Net and Recurrent U-Net. It introduces recurrent convolutional layers and residual connections to the U-net architecture. The recurrent nodes introduce feedback loops which enable units to incorporate context information from adjoining units into their feature maps. This should improve feature representation, accuracy, and performance. R2U-Net combines the input of the first convolutional layer with the output of the second layer at each block using a skip connection. This allows for design on deeper U-Net's with recurrent nodes by preventing the vanishing gradient problem [1] [6].

R2U-Net beat the training loss of vanilla U-Net and had only a slightly higher validation loss than U-Net. This is a promising result and merits further investigation with training for a larger number of epochs. R2Unet's validation loss (0.288) was quite close to that of Unet (0.287).

3.3. Attention Unet

The Attention U-net architecture adds attention gates to the base U-net architecture. This allows the model to suppress irrelevant regions in the image while highlighting the salient features useful for the reconstruction. Attention gates are incorporated into each layer in the expansive path and cropped features from the skip connections must pass through them before they can be concatenated with the up sampled feature maps. This allows different parts of the network to focus on segmenting different objects since attention gates provide localized classification information [4] [6].

Attention U-Net did not perform as well as vanilla U-Net with both training and validation loss higher than the baseline. It appears that attention gates do not provide much separation between relevant and irrelevant regions in the image and the U-Net architecture does a better job of segmentation without them. Attention Unet had the best validation loss of 0.295.

3.4. Dense Unet

During the initial phase of the project, while we were exploring different available models, Dense Unet showed up as another popular adaptation of U-Net. As the name suggests, Dense Unet consists of dense layers/blocks and hence can prove to be quite demanding computationally. We came across several distinct applications of Dense Unet architecture in [2]. In the field of Neural Networks Dense Unet is also known as the BarbellNett extension. Essentially it is a fully convolutional neural network architecture that utilizes the channel attention mechanism using the residual channel attention block (RCAB).

The way we implemented it, our architecture is an encoder-decoder setup similar to Unet but uses the residual channel attention block as its key module. In the encoder phase, the input feature maps are pooled with strided convolution to reduce the computational load and increase the receptive field size. The feature maps are then passed through a feature extraction layer of multiple RCAB modules. Finally, the feature maps are up-scaled and concatenated with feature maps from the encoder stage. We specifically used the Dense Head BarbellNett flavor which uses a dense block for feature extraction of the inputs. The model outline for our Dense U-Net implementation can be found in our repository. In our initial attempt to train the Dense Unet, we realized we would never finish in time given the high computation demand. Hence, we had to reduce the number of parameters which eventually did not impact much in terms of validation loss. Dense Unet is commonly known to reserve much more detail in the image samples but simultaneously it is also known to sometimes result in over smoothening of the images just like Unet. The best validation loss for Dense U-Net was 0.292 vs 0.287 for U-Net.

3.5. Nested Unet

The paper for Nested Unet's turned out to be quite popular as we could easily find a bunch of enactments of the same by a lot of people. Just by going through the paper [8] we were sure that this will prove to be a very successful and

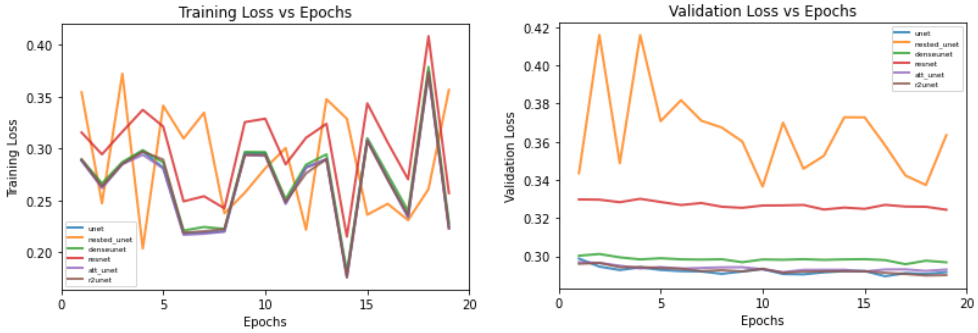

Figure 2. Training and Validation Loss

powerful architecture for medical image analysis. Nested Unet basically is a deeply supervised encoder-decoder network where the encoder and decoder sub-networks are attached through a series of nested and dense skip pathways. The re-designed skip pathways target at decreasing the semantic gap between the feature maps of the encoder and decoder sub-networks. The primary difference between Unet and Nested Unet is the re-designed skip pathways (figure 1 of [8]) that connect the two sub-networks and allows the use of deep supervision (shown in the same figure in red).

Segmenting abrasions or abnormalities in medical images calls for a higher level of accuracy than what is desired in natural images. That's where the skip connections have proved very effective - recovering fine-grained details of the target objects, generating segmentation masks with fine details even on complex background. Skip connections is also fundamental to the success of instance-level segmentation models, which enables the segmentation of occluded objects. Within the limited timeline and compute resources, we trained the nested unet on the dataset with deep supervision. The model trained very well and achieved one of the least training losses among all other models. Nested Unet is larger in comparison to U-Net (Figure 1), and takes much longer to train. Maybe if we had more time and better access to GPUs, we could have tried hyper-parameter tuning to try to get even better results with Nested Unet. The best validation loss for the Nested Unet was 0.333 vs 0.287 for U-Net.

3.6. Resnet

We found several different implementations of Resnet on different tech forums. Keeping in mind the compute resources we finalized the one described in this blog. We modified the model a bit, though. While doing our research, we found an interesting article here on stackoverflow about the effects of different styles of paddings in neural networks. We replaced the zero-padding implementation with reflection padding and it gave us better results in terms of validation loss. The comprehensive model summary of our ResNet implementation can be tracked in our repository and it can be seen that our model pads the outer edges with its own reflection. The image is then passed through a series of Conv2d, batch normalization and Relu layers – 6 times to be precise (with no Relu in the last layer). Comparatively, U-Net is a bigger model with far more layers than Resnet. Also, our implementations were altered specifically as per the limited compute resources at hand. The key comparisons between the various models are listed in Figure 1. When it comes to the validation loss, Resnet model had the validation loss of 0.321 as compared to U-net 0.297.

3.7. Image Reconstruction

This part of the project was impacted the most due to the loss of data caused by GCP Issues (mentioned in Challenges section of this report). The process of reconstructing the image was carried out on the test data set. The test data files had 36 slices of sampled data for a patient. Through hit and trial we discovered that Slice 22 has most complete appearance of a knee and hence is used for reconstruction. The final reconstructed images for all the models are shown in Figure 3. It can be observed that most of the images have inverted gray colors. We think the reason for this is that as we ran the training models for only 20 epochs, the weights might not have stabilized. We tried to debug the issue but because of the extremely shortened timelines (GCP Issues) we were not able to do the due diligence to this problem. The feature retention can still be noticed for different model's images and that is the reason we decided to show the images as is, to establish a proper comparison between the models.

4. Challenges

4.1. COVID Impact

Our team of 3 started working on the project quite early and with a lot of enthusiasm. We did our research, explored different project options and read through a plethora of papers to agree on a direction. But then one of our team

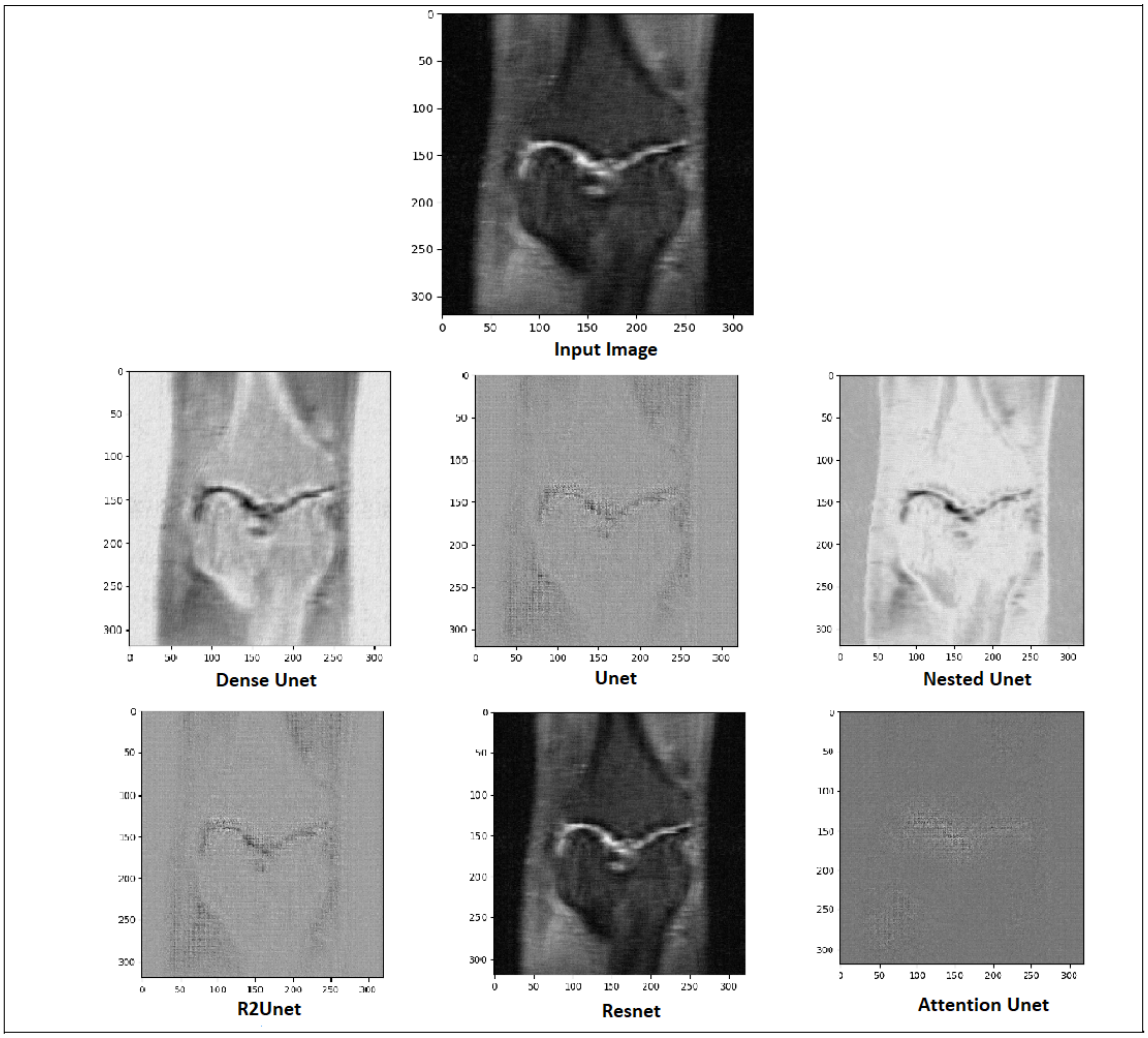

Figure 3. Image Reconstruction Results

mates Mohit Chawla (and his family) unfortunately caught Covid. Due to severe fever, weakness and other symptoms Mohit was unable to contribute towards the project from 4/8 through 4/22. The remaining teammates kicked off the project, but only to be impacted by another hurdle of cloud environment setup, which as per initial assignments was assigned to Mohit. Once Mohit was able to contribute, we spent countless hours on daily meetings and brought the project back on track, but we had to cut down on the scope. As a team we all we agreed to drop more complicated (potentially superior) models which meant that scope had to be restricted to implementation and comparison of U-Net inspired models to try and find results better than the state-of-the-art models or else get comparable results to Unet with computationally lighter models. We also had to abandon our plans for hyper-parameter tuning.

4.2. Google Cloud Platform (GCP) Issues

Due to back-to-back GCP issues in the penultimate week, we lost all our data and training progress for several models twice. Initially the US instances collapsed for the VM; we explored some European instances and spinned up a new VM. Two days later all regions went into a shut down mode with no option to set up a new VM in any region. We lost more than 4 days worth of progress and spent the next 2 days on exploring other cloud service providers.

All these issues are mentioned in detail in this private piazza post. Even thought the instructors were kind and supportive enough to allow us to cut down more on the scope if needed, we did not give up and explored other cloud services like AWS and Vast.AI. Finally, after spending 2 days we were able to set up new VMs from scratch on Vast.AI.

But this came at the expense of not sparing us with any time to debug any issues that would be found in the new

trainings executed for all the models on the Vast.AI platform. One of the issues being in Image reconstruction where the images formed in almost all our models had inverted colors. We tried spending a few hours on debugging the issue but had to transition our focus to the report, github and other last minute deliverables in order to successfully finish the project. Also, with the shortened timeline, we could train all the models for 20 Epochs only.

5. Work Division

Refer to Figure 4 (Page 8).

6. Conclusion

Results for all the models compared can be found in Figure 1. This project work presented a great learning opportunity to us. Having faced some unexpected roadblocks, we were still able to achieve most of what we had initially planned. We set out to find comparable models as Unet in terms of either a better validation loss or else a comparable validation loss but with lighter computational demands. If we look at the Figure 1 we can see that we found mix of both worlds.

We can see that Attention Unet and R2Unet are larger models but have comparable Validation losses almost as good as the Unet. Nested Unet is also not too far away in the category of computationally heavier models ending with slightly higher validation loss. If we talk about Dense Unet; we agree that we had to tweak Dense Unet a little to reduce its parameters or else given the dense nature of its layers we would have never finished the training part in time; but still it was able to deliver almost the same validation loss as Unet (with a difference of 0.01).

6.1. Learnings

Based on the results and the analysis, the more complex models based on the number and size of learnable parameters do not necessarily produce better results.

We also learnt that we might need to discover an extreme remodel of the underlying Unet model to break through the plateau of the validation losses (all models staying in the range of 0.28 to 0.33).

7. Future Research

Given more time, the first thing on our agenda would be to perform hyper-parameter tuning of the models in an attempt to bring down the losses. We had reserved the last week for this activity but since we were recovering badly from the data loss caused by GCP issues, we had to put this in the future research bucket. Another thing we would definitely like to try is to train the models for at least 50 Epochs. During our initial research, where we explored several tech journals and existing papers we observed that the common trend is to train the model for at least 50 Epochs. We completely understand this would require heavy computational resources and that's something we found an answer in Vast.AI. When we had to transition from GCP we found that Vast.AI was far more reliable and reasonable to hire more powerful GPUs. It was only for Vast.AI and the more powerful GPUs they offered that we could finish running all models from scratch in the last week. This would even let us test more combinations of hyper-parameters. Third, we aspire to fix our inverted colors issue with the image reconstruction part. We were able to get some decent images for some of the models and would be an icing on the cake to fix the color space issue. Lastly, we would like to finish our research for other models that we had initially planned but had to cut down on the scope due to Covid impacts and GCP issues. We were working on some very interesting other models like the one where we would combine the KNN (nearest neighbors) technique with the convolution model in the underlying Unet code-base. This has proved to be very successful in some of the results that we came across and that too with very light computation demands.

References

[1] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955*, 2018. 3

[2] Joonhyung Lee, Hyunjong Kim, HyungJin Chung, and Jong Chul Ye. Deep learning fast mri using channel attention in magnitude domain. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 917–920. IEEE, 2020. 3

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[4] 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

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2, 3

[6] Nahian Siddique, Paheding Sidike, Colin Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: theory and applications. *arXiv preprint arXiv:2011.01118*, 2020. 3

[7] Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, et al. fastmri: An open dataset and benchmarks for accelerated mri. *arXiv preprint arXiv:1811.08839*, 2018. 1, 2

[8] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018. 3, 4

Student name	Contributed Aspects	Details
Mohit Chawla	- Researched cloud provisioning and performance. - Set up and debugged cloud environments. - Primary responsible for model testing, image reconstruction. - Implemented Dense U-Net model. - Implemented graph creation code. - Wrote README and reviewed report. - GitHub repo creation and check-ins.	- Did research into setting up Cloud VMs on all major public clouds - Spinning up of cloud VMs, debugging GCP pre-empting, downloading data sets and managing file transfers to and from test environments - Ran training and inference on models including generating reconstructed images - Wrote notebook for creating test and validation plots and generated plots for report - Trimmed run log for plot generation and check-ins to GitHub - Wrote README file - Collated and checked-in all code, logs, plots, and images into GitHub
Vaneesh Bahl	- Researched models for implementation. - Researched hyper-parameter tuning and graph formation. - Implemented/trained ResNet and Nested UNet models. - Secondary responsible for testing. - Secondary responsible for Cloud VMs. - Wrote and finalized report, Overleaf setup.	- Read papers on image segmentation models and self-supervised learning applications. - Spun up VM instances on Google Cloud Platform (GCP). - Strategized Nested Unet and Resnet models implementation. - Debugged GCP training/inference issues. - Explored options for graph formation to track losses for various models. - Investigated prospects for hyperparameter tuning for computationally heavy models. - Configured the report in Overleaf and wrote the sections 3.3 onwards of the report.
Prashant Sharma	- Researched models for implementation. - Did project set-up and experiments with U-Net. - Implemented R2U-Net and Attention U-Net models. - Implemented code for image reconstruction. - Secondary responsible for testing. - Wrote report and README.	- Researched papers on biomedical image segmentation to select models for implementation - Set up the project by reviewing fastMRI codebase, requesting datasets, defining project structure, investigating environment setup issues - Did initial experiments with the baseline U-Net model to set up loss benchmarks - Researched and implemented the R2U-Net and Attention U-Net models and debugged testing issues - Wrote code for image reconstruction during inference - Wrote report - Wrote README

Figure 4. Team's contributions