FastMRI Project Report : CS 7643

Jaykumar Kakkad
jayk@gatech.edu

Priyank Sharma
psharma349@gatech.edu

Akit Singh
asingh822@gatech.edu

Mordecai Weisel
mweisel3@gatech.edu

Georgia Institute of Technology

Abstract

Over the last half century, MRIs have been a critical diagnostic tool for medical professionals, but has been plagued by high costs, both in terms of time and money. Work has been done to lower cost through software improvements. In the last decade, substantial improvements have been made through the use of deep learning. In this paper, we consider a number of approaches that have been suggested to solve this problem using deep neural networks.

1. Introduction

Magnetic Resonance Imaging, or MRI, is an important diagnostic tool in modern medicine. It is used to diagnose issues as varied as brain injuries, aneurysms, strokes [4]. An MRI is an image of the body that is produced by measuring radio frequency that is "pulsed" through the body; these signals are transformed into a 3D image of the part of the body that is being investigated [3]. These signals, known as the k-space, are processed as matrices of complex numbers that are manipulated to produce a 2-dimensional matrix of pixels that form the image. As will be discussed, despite software advancements, the current state of collecting the radio frequencies required for an MRI is long and expensive [4]. In this project we explore methods to make this process faster and cheaper by generating high resolution MRI images from radio-frequency signals that were collected over a shorter period of time than the current state of the art.

1.1. Who Cares?

Even though MRI is widely used, the long acquisition time of the order of 45 minutes makes MRI inconvenient for children and the elderly and also increases the cost of examination. To reduce the MRI time, researchers have developed compressed sensing methods. In general, it works by using less measurement but still reconstructing a quality image. It uses under-sampled inputs from K-space and uses a variety of algorithms to reconstruct the MRI images. Classical approach is to view this problem as an optimization of a regularized model.

$$\arg\min_{\alpha} \frac{1}{2}\|PFA\alpha - y\|_2^2 + \lambda\|\alpha\|_p$$

Where: P is undersampling, F is Fourier transform, A is the basis.

However, these classical methods have had limited success given the high accuracy required by the medical industry. MRI's are still expensive and inconvenient. Speeding up the diagnostics process will help patients mentally and burden less on their pockets at the same time producing these results with accuracy will enable a reliable tool for medical professionals.

1.2. How is it done today?

MRI scanners acquire collections of Fourier frequency "lines", commonly referred to as k-space data. Due to hardware constraints on how magnetic fields can be manipulated, the rate at which these lines are acquired is fixed, which results in relatively long scan times and has negative implications with regard to image quality, patient discomfort, and accessibility. The major way to decrease scan acquisition time is to decrease the amount of data acquired. Based on sampling theory there's a minimum number of lines required for image reconstruction which can be circumvented by incorporating other techniques such as parallel imaging and compressed sensing. More recently, machine learning methods have demonstrated further accelerations over parallel imaging and compressed sensing methods. Most of these models use the U-Net based variational networks or some form of CNN architecture [10]. More recently, there has also been mention of using reinforcement learning [11] to personalize AI-accelerated MRI scans. All these efforts try to speed up the MRI scans by reconstructing the scan using fewer measurements.

1.3. Dataset

For the purpose of our experiments we used the fastMRI dataset that contains four types of data from MRI acquisitions of knees and brains [2]. We used the single-coil knee dataset for all of our experiments. The dataset is divided into four parts: training, validation, test and challenge. Volumes in the test and challenge datasets contain undersampled k-space data which is performed by retrospectively masking k-space lines from a fully-sampled acquisition. k-space lines are omitted only in the phase encoding direction, so as to simulate physically realizable accelerations in 2D data acquisitions.

Training and validation dataset contains fully sampled k-space data. For our experiment, we set the acceleration factor to four (representing a four-fold acceleration) with random masking. Ground truth images were not provided for the test and challenge dataset. Ground truth images for the training and validation set were constructed using Inverse Fourier Transform which was given as ESC reconstruction in the dataset. Due to the limitations on computational resources to train, we decided to use only 25% of the entire training set, while validation and testing is performed on the full dataset.

2. Approach

We implemented a total of five architectures. First implementation was the U-net architecture as a baseline. To increase the inductive bias, we improvised on this architecture by implementing Deep Residual U-net and U-net++ architectures. We then experimented with Vision transformers to see if capturing more global information can help improve the performance. Finally, we explored self-supervised learning using the GAN architectures given its recent successes [15].

Since we used U-net as our baseline, we leveraged the implementation and pipeline provided by the fastMRI repo [1]. For all other models, we changed the code construct to accommodate for any new architecture, loss function, training steps etc. while using the fastMRI pipeline to execute the dataloader and training loops.

2.1. U-net

CNNs are the most popular model to tackle vision problems and hence to tackle the problem of MRI reconstruction, a simple encoder decoder architecture was proposed by Ronneberger et al [13]. The encoder architecture consists of two 3x3 conv2d layers at each block followed by ReLU or Leaky ReLU activations with instance normalization and a 2x2 max pooling layer with stride of 2 to down-sample the image. At each down-sampling step, the number of channels is doubled. The decoder architecture consists of an 2x bilinear upsampling layer that halves the number of

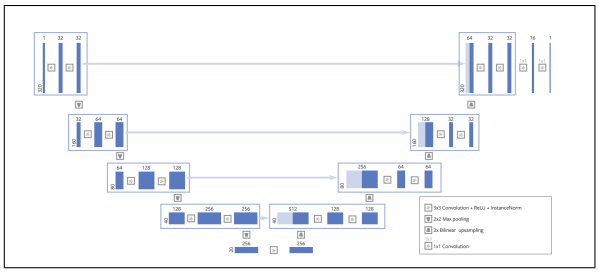

Figure 1. U-net architecture. Image from [13]

feature channels. After the upsampling layer we concatenate the output of the corresponding down sampling layer and then passed on to the conv2d block which consists of 2 3x3 conv2d layers followed by ReLU or leaky ReLU with instance normalization. At the final layer 1x1 conv2d layer is used to reduce the number of channels to one without changing the spatial resolution.

Base code for training U-net is provided by Facebook Research and hence we used that module to train U-net from scratch where RMSProp was used as optimizer with a learning rate of 0.001. We trained U-net on a subset of data and used this as base performance to evaluate other models. The model was able to learn quickly but saturated after 10-11 epochs. This is mainly due to small sample size provided to the network and was expected. Base U-net with 32 channels achieved 33.78 PSNR value on 4-fold acceleration while training the same network on only one fourth of the data was also able to learn reconstruction and achieved a score of 31.9 which is greater than classical method which achieved a score of 30.69.

2.2. Dense Residual CNN

This architecture mainly improves on the idea of skip connections of a U-net. The backbone structure is the traditional U-net, but the layers connecting Up-sampler and Down-sampler layers are modified to a dense CNN architecture. This idea was first mentioned in [14] as part of Dense U-net and further elaborated in [7].

Our idea to pursue after this architecture was driven from the incapability of U-nets to capture high frequency data due to the decoder and encoder framework. With this design our aim was to improve propagation of information from higher layers to lower layers. The skip block contains a dense layer (2 layers of CNNs with ReLU and BatchNorm) and has skip connections in between for residual propagation. This skip block is used across all pairs of Encoder and Decoder blocks. We first trained the model over default setting as the base U-net which led to instances of overfitting (seen by increase in validation loss while training loss continued to decline) and poor learnings, hence the next iteration was designed to optimize the hyperparameters for more

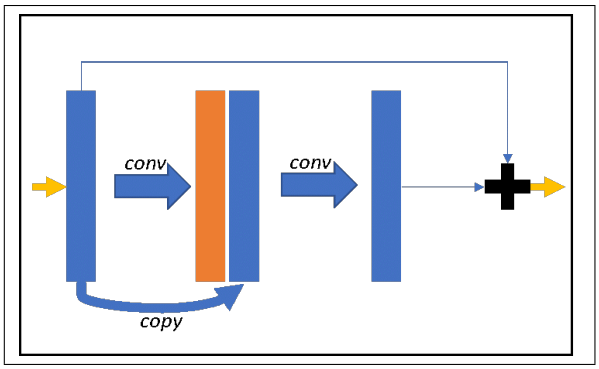

Figure 2. Dense Residual CNN architecture.

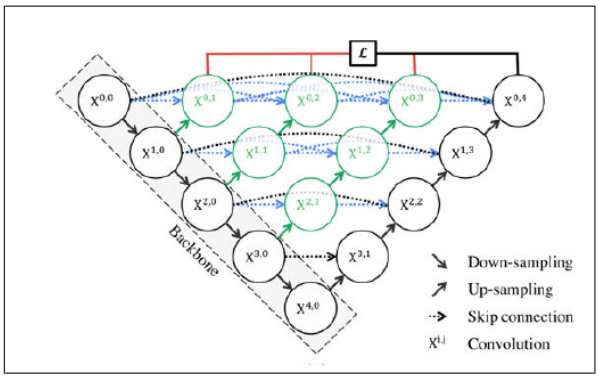

Figure 3. U-net++ architecture. Figure from [8]

effective results. We introduced a layer of dropout with 0.1 probability and reduced the size of layers in encoder and decoder architecture by 2x to ensure reliable training on small dataset. Figure below captures this learning behavior across the two variants of this dense architecture.

2.3. U-net++

We selected the U-net++ architecture to implement to further improve the skip connections that would improve the inductive bias exhibited by the vanilla U-net architecture. In the dense U-net architecture, we tried to improve the skip connection and further improve gradient flow. However, the encoder and decoder networks can be semantically different which could hinder the learning task. Hence, Zhou proposed a new architecture called U-net++ that reduces the semantic gap between the feature space of encoder and decoder [17]. U-net++ is based on U-net architecture. It differs in that the skip connections are designed as dense convolution layers. The number of convolution layers of each skip connection depends on a pyramid structure. The architecture also uses deep supervision by averaging loss values at various convolution layers. However, given our computational constraints we used only the loss value from the last layer also called fast mode. The paper uses a combination of dice coefficient and Binary cross entropy of predicted and actual image as loss function. However, cross entropy requires that the target has values between 0-1 (probability distribution) and our target was not in that range. Hence, we used only L1 distance between actual and predicted values as loss function. Given our computational constraints, we used Adam optimizer because it is computationally more efficient compared to other methods. Based on lower data, we decided to reduce the model capacity and use only starting channel size=16. After a few epochs, we found that the validation loss did not decline but the training loss kept reducing. Hence, we decided to increase regularization and hence set drop-out at 10%. Even with these parameters, the model still overfits probably because model capacity is still too high for the data points.

2.4. Visual Transformer (ViT)

In this experiment we expanded our study beyond Encoder-Decoder models towards the Vision Transformer (ViT) for accelerated magnetic resonance imaging. Since the introduction of Vision Transformers and their revolutionary success in computer vision [8], a number of recent works have proposed to utilize Transformers or self-attention mechanisms also in image reconstruction methods. We implement adapted ViT proposed by Lin et al [9].

The Vision Transformer(ViT) is similar to the original transformer encoder used for processing sequential data like speech. ViT first converts an image into a sequence of images of equal size(patch embed) and then uses this as a sequence of data. Since there is no positional information associated with the images, we introduced positional embeddings associated with each patch of image. These position embeddings are d-dimensional vectors, which encode information about the absolute position of sequence elements. The image patches are then converted into a d-dimensional feature vector using conv2d on patches with stride and kernel size equal to the size of image patch with number of channels as d(64 in our case). These two d-dimensional vectors are added and passed to the transformer encoder.

Transformer encoder consists of 8 encoder layers, where each layer contains a multi-head self-attention(MHSA) block followed by a two layer MLP block with layer normalization and residual connection after each block of MHSA. Each MHSA block consists of Query vector(Q), Key vector(K) and value vector(V), which are learn-able parameters and help to calculate attentions using the equation

$$attn = softmax(\frac{(Q * KT)}{\sqrt{d}}) * V$$

where KT represents the transpose of K. All the attentions are then concatenated together and passed on to the linear layer to transform it into a vector. The output vector

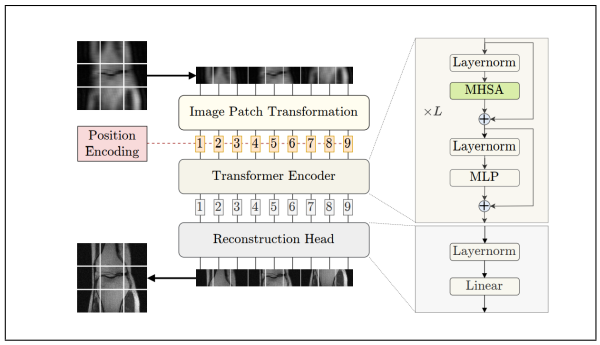

Figure 4. ViT architecture. Figure from [9]

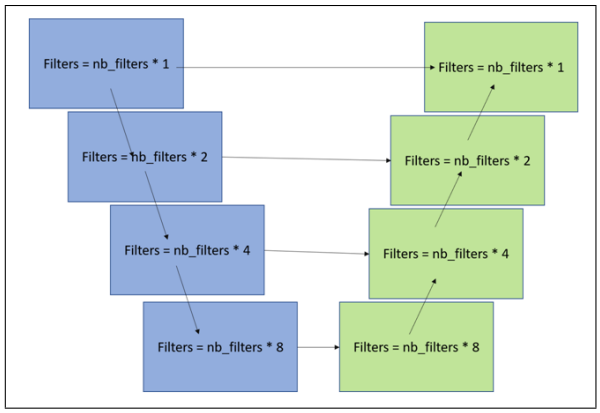

Figure 5. ReconGAN architecture.

is then passed on to the MLP block which consists of two linear blocks with a GELU activation in between.

The output from the transformer encoder is passed on to the reconstruction head, which comprises of a layernorm followed by a linear layer which transforms the output into shape of (B, N, H* W), where B is batch , N is number of patches and (H,W) is the shape of image patch, which is then converted back to image space. We then compare this reconstruction with the ground truth using L1 Loss. One thing we observed was writing reconstruction of image space from tensor can be rearranged. Rearranging in 1st dimension that belongs to number of patches doesn't affect the image reconstruction because it can be learned through previous linear layer through proper gradient flow.

2.5. General Adversarial Model (GAN)

The problem of generating high resolution images from under-sampled k-spaces can be modeled as a self-supervised one by training a model to generate images based on k-space input, and use the true image that was generated by the k-space sample as a basis for the error. In order to simulate under-sampled k-space, we apply masks to the k-space samples, so that the model learns to generate a high-resolution image from k-space samples as a means for the model to measure whether it is generating a realistic image based on the input k-space. General Adversarial Networks (GANs) have been shown to perform well in this space [16]. We implement the ReconGAN proposed by Quan et al [12]. This architecture is characterized by two primary components. The first is a generator model, which takes as input an image that was created from mask k-samples, and generates another image. The second component is a discriminator model, which takes as input an image, and determines if that image is a real image or generated.

The generator takes the form of an autoencoder. The autoencoder is composed of eight blocks, four for the encoder, and four for the decoder. Each block is composed of six CNN layers (or Transposed CNN layers in the case of the decoder). All layers have a kernel size of 3. The first layer of each block has a stride of 2 and padding of 1. The distinguishing factor between the blocks is the number of filters.

The filters of the layers are some multiple of a hyperparameter nb_blocks, which we set to 64, except for the fourth and fifth CNN layers, which have half the number of filters as the other layers in its block. The input into each block of the encoder is the preceding encoder block. The input into each decoder layer is the preceding decoder block, as well as the corresponding encoder block. This architecture, similar to U-net, is meant to make use of skip connections to ensure proper gradient flow throughout the network. It has an encoder element series of CNN blocks that down-sample the image, and a corresponding decoder element, which is a series of CNN blocks that up-sample the product of the encoder.

In addition to backpropagating the discriminator's loss to the generator, the generator is trained with its own loss function. The function measures the difference between the generated image and the true image, as well as the generated k-space and the true k-space. This loss function, which Quan et al call Cyclical loss [12], is meant to ensure that the generator produces the specific image that should be produced by the input k-space. If it were trained based on the discriminator alone, the generator would learn to generate realistic images, but not necessarily the image that should result from a given k-space. The discriminator takes an image as input, and outputs the probability that the picture is real or model-generated. It begins with an identical architecture to the encoder section of the generator. Its last layer is a CNN convolution, outputting a down-sampled version of the encoded image. The loss function of the discriminator is the difference between the sigmoid cross entropy loss of the real image (where the target value is 1) and the sigmoid cross entropy loss of the generated image (where the target value is 0). Because of the adversarial nature of the generator and discriminator, training can be unstable. To

Figure 6. SSIM and validation loss Dense Residual CNN(Cyan and Dark blue), Unet++(Red) and BaseUnet(Orange).

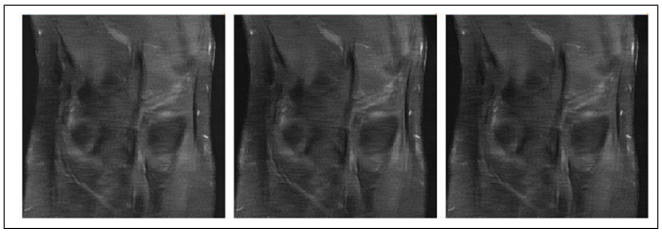

Figure 7. Reconstructed images from Desne Residual CNN, Unet++ and Target (left to right)

address this problem, Wasserstein GAN energy was used [6]. A bug in the model implementation caused the generator to train more often than the discriminator. This led to a generator that was able to fool the discriminator by generating an output that did not look like a real image, but the discriminator was too poorly trained to properly identify it as such.

3. Experiments

As noted above, all models were trained on 25% of the training data, and the entirety of the validation and test sets. The hyperparameters and loss functions vary from model to model, and are described below. In order to compare between models, we used their SSIM score as a quantitative metric, and visually inspected the produced images as a qualitative metric. The SSIM score, proposed by Wang et al, was chosen because it measures "local patterns of pixel intensity," that is to say, it compares the structures of the true image and baseline image [15].

As can be seen from Figure 7,8 and 11, all models aside from ReconGAN were successful in producing a realistic image. As will be discussed in greater detail, these models were able to train well enough to produce realistic MRI images when inspected visually. However, none of our models, other than the Dense CNN was able to reach the SSIM score achieved by the baseline U-net model. In the subsequent sections we will detail why these models fell short of the baseline.

3.1. Dense CNN

While working through Dense residual CNN architecture we could observe better performance compared to base U-net architecture over smaller epochs but in longer epochs base U-net outperformed on all parameters. Figure 6 captures different performance metrics to highlight this. Major cause of this behavior can be attributed to the scope of hyperparameter optimization of dense residual CNN architecture. In our experiment we could optimize the hyperparameter based on heuristics due to 8hr long run time for one training iteration and limited resources. However, we feel executing more variations of hyperparameter optimization could have led to better results. Despite the above results it is evident that under small training steps dense residual CNN skip layers perform better than direct skip connections. Also, upon visualizing the reconstructed images as part of Figure 7 we could observe comparable performance with Base U-net but with clear demarcation in optimized architecture.

3.2. U-net++

The implementation did not yield results better than the baseline (U-net) or even the Residual dense CNN. The main reason could be significant increase in model capacity and not enough data to train it. Moreover, the validation loss stagnated and training loss kept declining which shows overfit of the model to the data. Hence, we believe the overall performance was marred by hyper-parameter tuning, higher model capacity and less data to train it. Visualizing the reconstructed images shows weak performance with respect to baseline and Residual CNN in Figure 7

3.3. ViT

ViT was trained for 30 epochs, the learning rate was set to .0005, and RMSProp optimizers were used for training. SSIM was used to evaluate the images produced by the network. The network training was originally triggered on 25% of data, but due to less credits, we were not able to download the output results from VM. Hence we repeated the experiment with only 70 slices to differentiate between CNN and attention model. With this experiment, we observed two things. First, even though both the models uses same loss function, ViT started with very large error, while U-net started with relatively less error suggesting that CNNs with random initialization also can extract some spatial information while attentions need to be learned based on the distribution. This shows that Vit needs more data to get correct attentions. Further, the learning curves were quite steep for ViT as compared to the U-net as seen in Fig 8. While U-net stops learning early, ViT needs more epochs to adjust the weights in order to perform well. In conclusion, if ViT was trained longer with sufficient data, it can achieve Base-Unet performance or even exceed it.

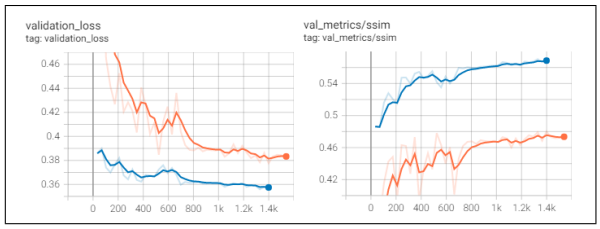

Figure 8. Base-U-net(blue),ViT(orange) SSIM and Validation loss. Both the models SSIM increased indicating that similarity was increasing while validation loss is decreasing indicating that both the models were learning

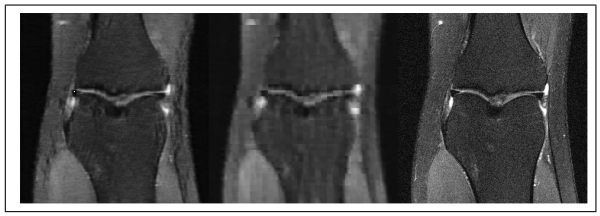

Figure 9. The original image on the right and Unet reconstruction on the left and ViT reconstruction in middle. This shows CNNs were able to learn quite easily while Vit struggled with small dataset.

3.4. ReconGAN

It was trained for 15 epochs, the learning rate was set to .0001, and RMSProp optimizers were used for both the generator, and discriminator. The generator and discriminator were trained by alternating between them, with the generator training every 5 timesteps. While code leveraged the U-net pipeline example provided by the fastMRI repo [1], since the majority of the example pipeline (except for the model and training steps) could be reused. The Pytorch Lightning module tutorial for implementing GANs was used as a reference for adapting the fastMRI pipeline to enable training a GANs type architecture [5]. The model failed to generate realistic MRI images, which more closely resembles the mask applied to the k-space than the k-space itself. As can be seen, the SSIM loss steadily decreased over time, although was orders of magnitude larger than the other models. Further, the model suffered from uneven validation loss, which began to diverge at the end of training. These findings suggest that the generator was not given enough epochs to train. It is possible that the SSIM loss would have continued to drop with either additional epochs, or training the generator more frequently than every 5 timesteps. The divergent validation loss suggests that the model may have suffered from mode collapse. Since the training data was only 25% of the original dataset, it seems that it may have performed better for those validation samples that were closer to the training samples, but diverged for those samples that were different. It should

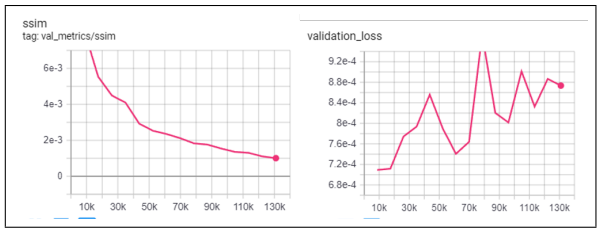

Figure 10. ReconGAN SSIM and Validation loss. While SSIM continues to decrease, Validation loss did not exhibit a consistant trajectory. These results must be plotted seperately from the other models because the dependent axis is orders of magnitude larger.

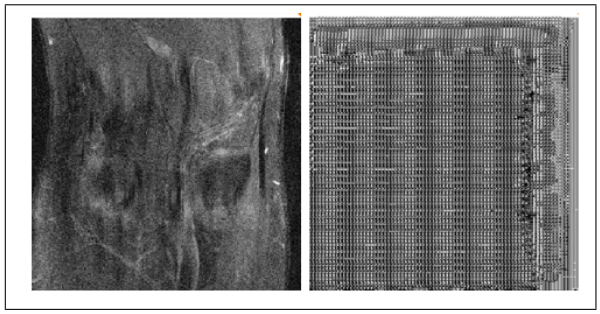

Figure 11. The original image on the left and GAN reconstruction on the right. The generator failed to learn to generate realistic images.

additionally be noted that since the generator model did not yet converge at the end of training, the discriminator was .0034 loss at the end of training, since it was still trivial for the discriminator to distinguish between true and generated images. It is likely that it would have risen if the generator model was allowed to continue training.

4. Challenges

4.1. Setup

It took a lot of time to set-up the VM infrastructure as getting the right combination of packages and VM was challenging. We used images provided by GCP and the latest T4 Nvidia VM.

4.2. Data

Since the training data size was 88GB, it would take an estimated 3 days to train the baseline. Hence, we reduced the size to 25% of the data

4.3. Baseline Execution

A substantial amount and time and resources were dedicated to executed the baseline code in the cloud. Poorly documented known-bugs in open source packages (Pytorch and h5py) resulted in memory leaks that rendered these executions useless. []

Student Name	Contributed Aspects	Details
Jaykumar Kakkad	Infrastructure, Residual CNN and U-net++, GAN, Report	Set-up the entire Cloud VM infra and related installations. Implemented and trained U-net++ models and contributed to the implementation of Residual CNN. Conceptual support in creating the GAN model. Writing my part and collating the details from other members
Priyank Sharma	Residual CNN and U-net++ architectures, Tensorboard	Implemented and trained Residual CNN models and contributed to the implementation of U-net++. Helped evaluation across models with results from Tensorboard and writeup for all of the above
Mordecai Weisel	ReconGAN (including custom training pipeline)	Implemented and trained ReconGAN model (including Lightning Module for unique training loop). Provided analysis for ReconGAN and edited write-up.
Ankit Singh	Training Pipeline,Base Unet, Vision Transformer(ViT)	Setup of Base training pipeline. Trained Base Unet from scratch. Preparation of small dataset. Implemented and trained ViT model. Conducted experiments to compare Unet and Vit. Write-up for section 1.3,2.1,2.4,3.3.

Table 1. Contributions of team members.

References

[1] facebookresearch/fastMRI: A large-scale dataset of both raw MRI measurements and clinical MRI images. available at https://github.com/facebookresearch/fastMRI as of August 4, 2022.

[2] http://fastmri.med.nyu.edu/ Available as of 8/4/2022.

[3] Magnetic Resonance Imaging (MRI) - available at https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri as of August 4, 2022.

[4] MRI - Mayo Clinic - available at https://www.mayoclinic.org/tests-procedures/mri/about/pac-20384768 as of August 4, 2022.

[5] PyTorch Lightning Basic GAN Tutorial — PyTorch Lightning 1.7.0 documentation - available at https://pytorch-lightning.readthedocs.io/en/stable/notebooks/lightning_examples/basic-gan.html as of August 4, 2022.

[6] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv*, 2017.

[7] Pak Lun Kevin Ding, Zhiqiang Li, Yuxiang Zhou, and Baoxin Li. Deep residual dense U-Net for resolution enhancement in accelerated MRI acquisition. 2019.

[8] Mage Is, Ransformers For, and Ecognition At. An image is worth 16x16 words. *The International Conference on Learning Representations*, 2021.

[9] Kang Lin and Reinhard Heckel. Vision Transformers Enable Fast and Robust Accelerated MRI. Technical report, 2022.

[10] Matthew J. Muckley, Bruno Riemenschneider, Alireza Radmanesh, Sunwoo Kim, Geunu Jeong, Jingyu Ko, Yohan Jun, Hyungseob Shin, Dosik Hwang, Mahmoud Mostapha, Simon Arberet, Dominik Nickel, Zaccharie Ramzi, Philippe Ciuciu, Jean Luc Starck, Jonas Teuwen, Dimitrios Karkalousos, Chaoping Zhang, Anuroop Sriram, Zhengnan Huang, Nafissa Yakubova, Yvonne W. Lui, and Florian Knoll. Results of the 2020 fastMRI Challenge for Machine Learning MR Image Reconstruction, 2021.

[11] Luis Pineda, Sumana Basu, Adriana Romero, Roberto Calandra, and Michal Drozdzal. Active MR k-space Sampling with Reinforcement Learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12262 LNCS, 2020.

[12] Tran Minh Quan, Thanh Nguyen-Duc, and Won Ki Jeong. Compressed Sensing MRI Reconstruction Using a Generative Adversarial Network With a Cyclic Loss. *IEEE Transactions on Medical Imaging*, 37(6), 2018.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, 2015.

[14] Nahian Siddique, Sidike Paheding, Colin P. Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, 2021.

[15] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

[16] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58, 2019.

[17] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11045 LNCS, 2018.