Reconstructing 9.4T MRI from 7T MRI images using Generative Adversarial Networks

Sadhana Ravikumar
sadhanar@seas.upenn.edu

Xuchen Wang
xuchenw@seas.upenn.edu

Wendy Feng
wendyfyx@seas.upenn.edu

John Bernabei
johnbe@seas.upenn.edu

Abstract

High-field MRI offers superior resolution and contrast compared to lower field strength MRI. However, high-field MRI is not as easily available and more costly. Therefore, a computational method for increasing resolution and contrast in medical imaging would be of significant medical and scientific importance. Here, we show that a generative adversarial network can be trained to transform 7T images to look more like 9.4T images by improving the contrast and resolution of the original images. We also show that this method performs superior to a fully convolutional network without a paired discriminator as well as a non-deep learning random forest model.

1. Introduction

High-field MRI scans provide higher resolution and better tissue contrast compared to lower field strength scans. With better resolution and tissue contrast, fine anatomical features can be seen more clearly, and post-processing tasks such as tissue segmentation and quantitative image analysis can be done more accurately. Unfortunately, high field MRI scans are more expensive, and not as easily available.

Previous work related to this topic of image-to-image translation has been done to reconstruct 7T MRI from 3T MRI. In this work, our goal is to develop and evaluate deep learning solutions to synthesize high field strength scans acquired at 9.4T from low field strength scans acquired at 7T. We will use a dataset of post mortem MRI scans of human brain specimens that have been scanned at both field strengths with an extremely high-resolution. Image synthesis of 9.4T MRI from 7T MRI scans would allow us to obtain better quality and higher-resolution images, while avoiding the additional costs and limitations associated with acquiring real 9.4T MRI images.

2. Related Work

Image-to-image translation is a very challenging problem since the mapping from source image to target image is non-linear and high-dimensional. Recently, several researchers have focused on addressing the challenge of estimating one modality image from another [13, 2] and single image super-resolution (SSIR) of MRI scans [16, 3]. Learning-based methods such as non-linear regression using random forest has been explored to perform image synthesis [7]. This approach relies on extracting a set of features from the source image dataset and then non-linearly mapping these features to generate the target image. The performance of this method is heavily dependent on the quality of the extracted source features.

Recently, deep learning has gained popularity in medical image analysis, and achieved great success in image super-resolution and synthesis without the need for hand-crafted features. Super resolution is the process of upscaling and or improving the details within an image. Dong *et al*. [4] propose using a convolutional neural network (CNN) for SISR. Given an input low resolution image, an end-to-end network is trained to learn the non-linear mapping to output the corresponding high resolution image. Kim *et al*. propose using a deeply recursive CNN to boost network performance and show that deeper networks achieve better super-resolution [9]. In [2], Bahrami *et al*. adopted a CNN to generate 7T MRI images from 3T MRI images by feeding the CNN with both the MRI intensity image as well as the corresponding anatomical segmentation. While using both appearance and anatomical features ensures anatomical consistency, the disadvantage of this method is the reliance on manually labelled anatomical segmentations. Labelled images are time-consuming to generate and not easily available for large datasets. When training CNNs, typically the mean square error (MSE) between the reconstructed and target image is used as the loss function. CNNs have been shown to be able to recover the structural details of high resolution images. However, the synthetic images generated using a CNN are

often blurry due to a lack of spatial context from neighboring voxels.

More recently, conditional generative adversarial networks (GANs) have been proposed for addressing this problem. In addition to training a convolutional generator network using adversarial learning, these networks simultaneously train a discriminator network which drives the generator's output to appear more similar to the ground-truth target image perceptually. In [13], a supervised GAN is trained to learn the mapping from 3T MRI to 7T MRI. By adopting an adversarial approach, they addressed the problem of blurry target images. In this work, Nie *et al*. propose including an additional image gradient difference loss in addition to the reconstruction loss and generator loss, to minimize the difference of the gradients between ground truth and synthetic images. The intuition behind this loss is to keep the strong gradients in the synthetic images with more prominent edges. They also adopt a residual learning framework to facilitate training the network. The Pix2Pix conditional GAN [6] is another popularly used approach for training a convolutional network for image-to-image translation tasks. It has been applied to a wide range of image translation tasks, such as converting maps to satellite photographs and converting black and white photographs to color.

3. Hypothesis

While CNNs can been used to generate synthetic 9.4T MRI from 7T MRI using a reconstruction loss, we hypothesize that incorporating a discriminator network on top of a CNN model will help the network generate better quality synthetic 9.4T images. The discriminator loss takes into account the network's ability to distinguish real and synthetic 9.4T images. This adversarial learning will provide the generator more information regarding subtle anatomic details in the 9.4T scans that can be used during the reconstruction process to fool the discriminator.

4. Inductive Biases

In this project, we will be using paired 7T and 9.4T MRI scans to train the different networks. Therefore, we know that the anatomical brain structure (context features) is the same across each pair of training images. We can therefore focus on training the network to learn the residual difference between the two images, which would be the high resolution details and contrast differences. We also know that there is a strong intensity gradient along the boundaries of anatomical structures that can be used to guide the reconstruction process. We believe that by incorporating a gradient loss function in the generator, this contrast could help inform detailed edges within the image.

5. MRI Dataset

The dataset consists of 3D post mortem scans of the medial temporal lobe region of the brain from 44 specimens acquired at both 7T and 9.4T. The 9.4T scans were acquired with a resolution of 0.2 x 0.2 x 0.2 mm^3, and the 7T scans were acquired with a resolution of 0.4 x 0.4 x 0.4 mm^3. It is important to note post mortem scanning of the brain allows for a much higher resolution in general, compared to standard clinical in vivo MRI scans which are typically acquired with a resolution of 1 mm^3. In this work, we will be performing image synthesis using 2D image slices which can be obtained by simply slicing each 3D dataset along multiple axes.

5.1. Data Pre-processing

As part of the image pre-processing, both the 7T and 9.4T scans were corrected for bias field non-uniformity using the N4ITK algorithm [15] and normalized to a common intensity range by clipping the intensities below the 0.1 and above the 99.9 percentile and scaling the intensity range to [0,1000]. Following image acquisition, the images are not aligned with each other across both field strengths. To align matching specimens, the 7T scans were first up-sampled (by a factor of 2) to have same dimensions as the higher resolution, 9.4T scans. The two images were then co-registered using a combination of affine and deformable image registration. Figure 1 shows 2D slices of paired 7T and 9.4T scans, following image registration. It an be noticed that the 9.4T scans have a higher resolution, better contrast and sharper edges.

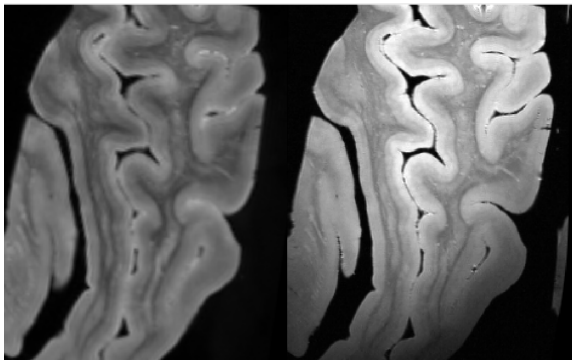


Figure 1: Example 2D slice of a post mortem MRI scan showing the 7T scan (left) and paired 9.4T scan (right)

5.2. Generating Patches

Due to the large and variable size of the input images, the networks were trained on image patches. As a first step, the dataset was split into training, validation and testing datasets using a 70/20/10 split. Each image was first

normalized to the range [0,1] and standardized to have zero mean and unit variance. To generate patches for training, in each specimen 300 corresponding patches of size 64x64 were randomly sampled across the paired image volumes. This resulted in a total of 9300 pairs of image patches to train the network, sampled from 31 different specimens. To generate patches testing and validation, the images were densely sampled, using a spacing of 20 voxels. At test time, the generated patches are stitched back together. Overlapping regions are averaged to obtain the final image. Patches of size 32x32 were also generated to assess how patch size affects network performance. Example input image patches are shown in Figure 2. From image pair shown in Figure 2-4, it can be noted that the 7T images may sometimes contain image noise that is not present at 9.4T. Furthermore, from the darkened 9.4T scan in Figure 2-5 we can see that MRI scans are prone to intensity inhomogeneities and other artifacts that are not necessarily consistent across both protocols. These issues make the problem of image reconstruction more challenging.

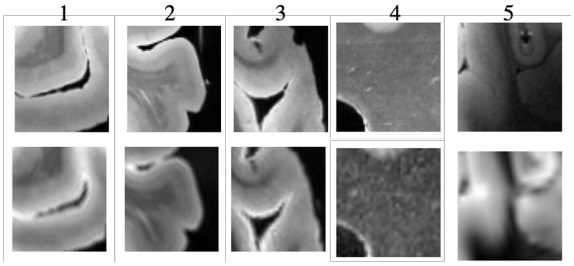


Figure 2: Example 2D 64x64 patches of 7T (bottom) and 9.4T (top) image pairs used to train the networks.

6. Method

6.1. Random Forest Super-resolution (RFSR)

As a non deep learning baseline, we used the random forest super resolution (RFSR) algorithm proposed by Shermeyer *et al*. [14]. We adapted the implementation from the authors to our dataset [1]. Training data for RFSR was created by downgrading the high resolution images and creating low resolution (LR) and high resolution (HR) image pairs. The LR images are shifted by a dimension of 2 in all directions, and each shifted images are stacked together then padded with zeros. RFSR trains on residual images - subtracting the actual LR images from this stacked LR images to create inputs to the Random Forest model, and from the HR images to create the targets. Subtracting the LR images removes the homogeneous areas, so that the model can focus more on the difference between LR and HR images.

[1] https://github.com/jshermeyer/RFSR

The scikit-learn Random Forest regressor was used to train on the residual stacked LR images, and residual HR images. The number of estimators (number of trees in the forest) used is 200, each with a maximum depth of 15.

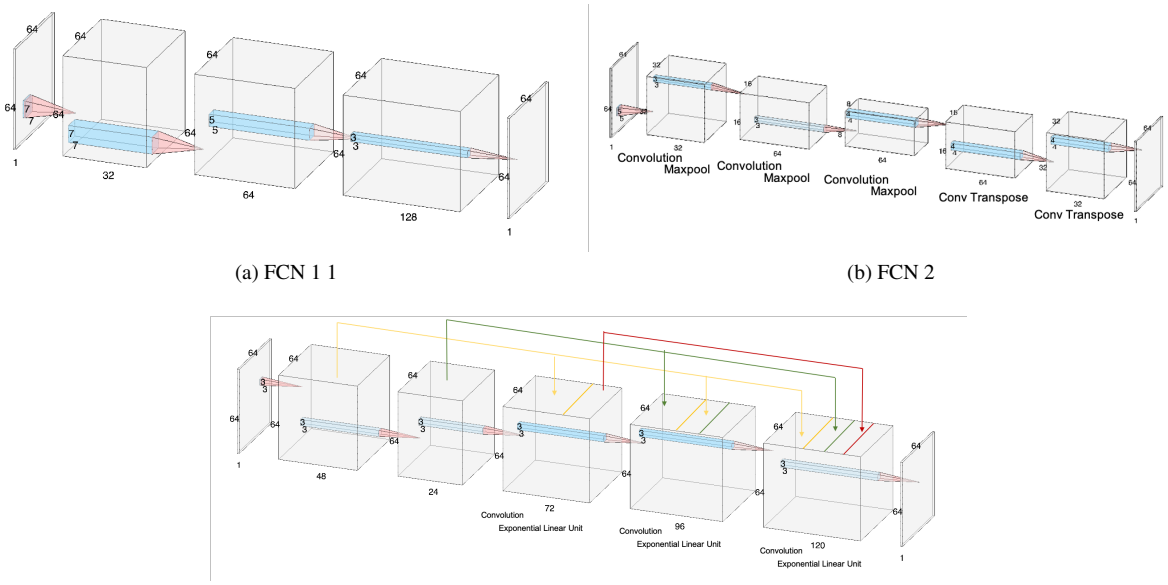
6.2. Convolutional Neural Network (CNN)

While mostly GANs have been used in image reconstruction problems, we were interested in assessing the effectiveness of a fully convolutional network (FCN) when performing this task to provide a deep learning baseline. In our initial experiments, a convolutional network, similar to the model proposed by Bahrami *et al*. [2], was used (FCN 1). This network represents the most basic form of a convolutional network and consists of 4 convolutional layers, each of which followed by batch normalization and a rectified linear unit (ReLU). Since this model preserves the input image size across all the network layers, we experimented with a second FCN architecture (FCN 2), which instead first down-samples and then up-samples the input image. Unlike the U-Net architecture, this network does not contain cross-connections between the encoder and decoder. Max-pool layers were used to down-sample the images in the encoder, and transposed convolutions were used to up-sample the image back to the input image size.

Lastly, we experimented with densely connected convolutional networks. Specifically, the densely-connected super-resolution network (DCSRN) described in [3] was implemented since it achieved good results when applied to brain MRI super-resolution. The first layer of this network is a convolutional later with a kernel size of 3 and 24 filters. This is followed by a densely connected block wich consists of 4 units. Each unit consists of a batch normalization layer, an exponential linear unit activation and a convolutional layer, with filter numbers increasing in each unit by a factor of 24. At the input to each densely connected unit, the output images from previous filters are concatenated. A final convolutional layer is used at the end to provide the final HR output. The network archiectures for the different CNN architectures explored (FCN 1, FCN 2 and DCSRN) are shown in Figure 3.

6.3. Generative Adversarial Network (GAN)

The conditional GAN model served as our advanced deep learning approach, to learn the mapping from the input images (low resolution images) to the output images (high resolution images). The generator is trained to generate fake images which appear as real as possible, while the discriminator is trained to detect the 'fake' images generated by the generator. The architectures of the models we found to be the most representative during our training process, as well as their hyper-parameters, are summarized in Table 1. The input, output dimensions as well as kernel size are marked on the architectures, see Figure 13.



(a) FCN 1 1

(b) FCN 2

(c) DCSRN

Figure 3: CNN Model Architectures

Residual learning was used in the generator network proposed by Nie *et al*. [13], so we started by implementing residual learning in the generator. However we found that by adding the source LR image onto the learnt residual, the generated HR image appear blurry, contains fuzzy edges and noisy background from the source LR images. To alleviate this effect, we found that directly generating the HR images from the input LR images generates much better results.

We first tried a different number of layers in the generator and the discriminator and found that generators with layers less than five could not capture enough information from the low-resolution images and generators with more than six layers could not improve the performance as much as we expected, so we decided to use the generator with five layers as our baseline GAN model, see part (a) in Figure 13. Based on model 1, we altered the kernel size and the filter number in the generator, see part (c) in Figure 13, to create model 2. Intuitively, larger filter numbers would capture more patterns and larger kernel size corresponds to a larger receptive field and thus considers more context. In addition, model 2 has a 1×1 convolution layer in the end to downsample feature maps. With the same idea, we further developed model 3, which has six convolution layers, larger kernel sizes, and more feature maps in each layer. The same changes were also applied to the corresponding discriminators. Comparing model 1 and model 3, the discriminator in model 3 has more feature maps to discover more patterns and has MaxPool layers to reduce the number of parameters. Model 4, builds on top of model 3, where we add a gradient difference loss to the generator loss function, see Section 6.4.

In general, the generator in these deep convolutional GANs (DCGANs) use 5-6 standard convolution layers which extract the high-level features from the input image, and batch normalization layers which normalize the output images. We further experimented with a more complex generator: the U-net generator from pix2pix, proposed by Isola *et al*. [6]. U-net is an encoder-decoder system with skip connections. The purpose of the encoder-decoder system is to extract a high-level compact representation of the input images although the image may lose pixel-wise information during the compression. As described in [6], since the output image of the generator has the same under-laying structure, to circumvent the bottleneck structure and reserve the information from the input, we add skip connections to concatenate the layers before the bottleneck with the layers after the bottleneck. The detailed U-net architecture can be found in part (g) in Figure 13. We replaced our original DCGAN generator with the U-net generator while keeping the same DCGAN discriminator, see part (d) in Figure 13.

Additionally, we created two different sized sharpening kernels to sharpen the edges of the generated images in model 6:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & c \end{bmatrix} \quad \text{and} \quad \frac{-1}{256} * \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

. Both kernels emphasize the pixel at the center of the kernel, while de-emphasizing the pixels near the kernel boundaries. Since the sum of the matrix elements is always 1, the matrix convolution does not change anything in a flat area but enlarges any intensity differences along edges. The inputs are convolved with the sharpening kernel after being passed through the U-net generator. We expect that the sharpening kernel would reduce the blurriness of the generated images and sharpen any edges. We also tuned other hyper-parameters, such as the weighting of the different loss functions in the generator, and increasing the generator learning rate. The hyper-parameters can be found in the second section of the Table 1

Finally, we experimented with the original pix2pix architecture, adding a PatchGAN discriminator on top of the U-net generator. A PatchGAN splits the input image into several patches, runs the traditional convolution models on each of them, and averages the final results to determine whether the image is real or fake. The main difference between a PatchGAN and a regular discriminator is that the output of a regular discriminator is a binary indicator, 0 or 1, while the output of the PatchGAN is a convolution layer, each element of which corresponds to each small patch. The main reason for us to use PatchGAN, as stated in the pix2pix paper [6], is that the PatchGAN can alleviate the problem of blurry images by forcing the discriminator to only model high frequencies, like edges and details. The detailed hyper-parameters for the pix2pix model can be found in the last section of the Table 1.

6.4. Loss Functions

When training the GANs, the standard conditional GAN loss is used in all the models, computed using binary cross entropy loss, shown in Equation 1 and 2. Here X_{real} refers to the original HR images, and X_{fake} refers to the generated HR images.

$$L_D = \frac{1}{2n}\sum_{i=1}^{n}(L_{BCE}(D(X_{real}),1) + L_{BCE}(D(X_{fake}),0)) \tag{1}$$

$$L_{G_{BCE}} = \frac{1}{n}\sum_{i=1}^{n} L_{BCE}(D(X_{fake}),1) \tag{2}$$

In addition, we experimented with three other types of losses for the generator - a reconstruction loss, a gradient different loss, and a perceptual loss.

Reconstruction loss, see Equation 3, takes in the original HR and generated HR and compute the L1 or L2 loss between the two. We experimented with both L1 and L2 loss and found that L1 loss produces less blurry picture than L2 loss with better edges and borders. L1 loss performs better on image generation tasks than L2 loss, most likely because L1 loss is less likely to be stuck in local minima, according to previous work by Zhao *et al*. [18]. The pix2pix paper [6] claims that L1 loss forces low frequency correctness, meaning that it ensures that the larger structures of the images remain unchanged, so that the discriminator and other losses (such as gradient difference loss) can focus on ensuring higher frequency correctness, i.e. the area around the edges.

$$L_{G_{Recon}} = ||X_{real} - X_{fake}||_k \qquad k = \{1,2\} \tag{3}$$

We also use the gradient difference loss proposed by [13], see Equation 4. The implementation can be found in their Github repository [2]. This loss minimizes the difference between the gradient magnitude along each dimension of the input. The gradient is computed by convolving the original HR and generated HR images with 2 kernels $[[-1,1]]$ and $[[1],[-1]]$, of size 1 and 2×1 respectively. Then the loss is computed by summing squared difference between the two gradients.

$$L_{G_{GDL}} = ||\nabla X_{real_x}| - |\nabla X_{fake_x}||^2 + ||\nabla X_{real_y}| - |\nabla X_{fake_y}||^2 \tag{4}$$

$$L_G = \lambda_1 L_{G_{BCE}} + \lambda_2 L_{G_{Recon}} + \lambda_3 L_{G_{GDL}} + \lambda_4 L_{G_{Perceptual}} \tag{5}$$

Perceptual loss, proposed by Johnson *et al*. [8] has been used in style transform and super resolution application. To compute the perceptual loss, input pairs of real and fake images are passed through a pre-trained convolutional neural network. The high level representation of image features obtained from this network is then used to compute the loss. In our implementation, we first normalize both the original HR and generated HR images to zero mean and unit variance, pass them through a pretrained VGG16 network and extract feature maps from layer 4, 9, 16, and 23. We then compute L1 loss for outputs from each of the layers respectively and sum them together to get the final perceptual loss.

The final generator loss combines BCE loss, reconstruction loss, gradient different loss and perceptual loss. Values for λ_1 through λ_4 are hyperparameters, and the final tuned value can be found in Table 1.

6.5. Training

All of the networks were implemented in Pytorch and run using the GPUs provided on Google Colab. For both the CNNs and GANs, the Adam optimizer was used to minimize the loss between the network output HR images and the corresponding HR ground truth during training. For the convolutional networks, a batch size of 32 was used during training, with a learning rate of $1e^{-3}$. The training

[2] https://github.com/ginobilinie/medSynthesisV1

Model		Architecture		Generator Loss (weight)					Learning rate	
Name	#	Gen	Disc	BCE (λ_1)	Reconstruction (λ_2)		Gradient (λ_3)	Perceptual (λ_4)	Gen	Disc
					L1	L2				
DCGAN	1	(a)	(b)	1	1.5	/	1.5	/	5e-5	2e-5
	2	(c)	(d)	1	1.5	/	/	/	2e-4	2e-5
	3	(e)	(f)	1	1.5	/	/	/	2e-4	2e-5
	4	(e)	(f)	1	1.5	/	1.5	/	2e-4	2e-5
U-net Gen	5	(g)	(d)	1	1.5	/	2	/	2e-3	2e-4
	6	(g)*	(d)	1	1.5	/	2	/	2e-3	2e-4
	7	(g)	(d)	1	2	/	/	/	2e-3	2e-4
	8	(g)	(d)	1	1.5	/	2	/	2e-3	1e-3
pix2pix	9	(g)	(h)	1	1.5	/	/	/	5e-5	2e-5
	10	(g)	(h)	1	/	1.5	/	/	5e-5	2e-5
	11	(g)	(h)	1	1.5	/	/	1	5e-5	2e-5

* Input images are convolved with a sharpening kernel after being passed through the generator network.

Table 1: GAN Hyperparameters

loss converged after 10 epochs. For longer training periods, the validation loss began to increase, suggesting network over-fitting. The reported networks represent the final tuned hyper-parameters after experimenting with larger/smaller learning rates and various filter sizes. We also experimented with using input images with a smaller patch size of 32x32. However, we found that this reduced network performance. When training the GAN, the various hyperparameters and loss functions experimented with are shown in 1. These networks were trained with a batch size of 32 and for 30 epochs.

7. Analysis

For our analysis, we first evaluated the results by looking at the generated HR images quality during training. If the results were satisfactory, we applied the trained models to generate HR images for the test set, stitch up the generated HR patches and computed Peak signal-to-noise ratio (PSNR), mean squared error (MSE) and Structural Similarity Index (SSIM), between the generated HR and original HR on the fully assembled image. Given a pair of $m \times n$ images of original HR, denoted X, and the generated HR, denoted Y, the evaluation metrics are computed as below:

$$\text{MSE} = \frac{1}{mn}\sum_{i}^{m}\sum_{j}^{n}(X(i,j) - Y(i,j))^2 \tag{6}$$

$$\text{PSNR} = 20 * \log_{10}(\frac{255}{\sqrt{MSE}}) \tag{7}$$

$$\text{SSIM} = \frac{(2\mu_X\mu_Y + c_1)(2\sigma_{XY} + c_2)}{({\mu_X}^2 + {\mu_Y}^2 + c_1) + ({\sigma_X}^2 + {\sigma_Y}^2 + c_2)} \tag{8}$$

where μ_X is the mean of image X, ${\sigma_X}^2$ is the variance of image X and σ_{XY} is the covariance, and c_1 and c_2 are constants.

When evaluating the generated HR images, we check for clear smooth edges, intensity among non-edge areas and correctness compared to the actual LR and HR images (i.e. is there new information in the generated HR that is not found in the original LR and HR images?). In the assembled, stitched together images, it is also important to check for edge effect on the edges of the patches, consistency of image intensity (including background) among multiple patches in the same stitched image.

7.1. Random Forest Super-resolution (RFSR) Analysis

The random forest model serves as our non deep learning baseline. It took almost 2 hours for the random forest to train on 2000 out of the 9300 training patches, so we decided not to train on the full training set. After visualizing the training and test result, the random forest generated HR images are almost indistinguishable from the original LR images, except slightly matching the intensity of the LR images to the HR ones. An example from the test set is shown in Figure 4. Tuning hyperparameters did not improve the results. RFSR trains on residual images, and add the trained residual back onto the original LR images to create the generated HR images. From the residual, we can see that RFSR is unable to learn the HR features, especially around the high intensity features around the edges. Given long training time, the weak residual images, and existing noise in the LR images, the generated HR images are not

satisfactory by evaluating the image quality. We decided to move on with deep learning methods.

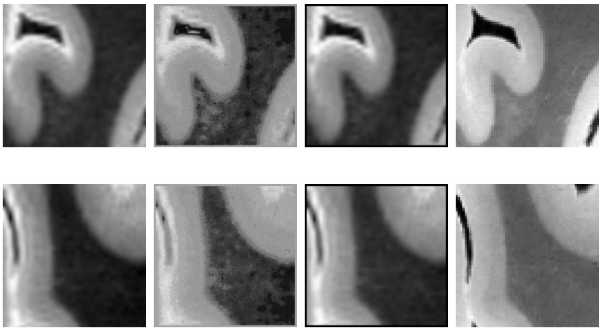


Figure 4: RFSR Example Test Output

7.2. Convolutional Neural Network (CNN) Analysis

Table 2 presents the evaluation results for the CNN models discussed above: FCN 1, FCN 2 and DCSRN. The DCSRN model achieves a significantly higher PSNR value, a higher SSIM value and lower MSE value compared to the other two architectures. Looking at the number of parameters present in each of the networks, the poor performance of FCN 1 and FCN 2 could be due to the large size of the network over-fitting the training data. The benefit of the densely connected network is that it results in less over-fitting of the training data since the number of parameters is greatly reduced. It also results in much faster training. By introducing skip connections, the densely connected model reuses features several times across the network. This makes it more difficult to overfit the training data. These effects can be observed in the training and validation curves shown in Figure 5 and Figure 6. The training loss converges much faster for the DCSRN model compared to FCN 1 and FCN 2. Furthermore, despite all three models converging to a low loss value, the validation loss for FCN 1 and 2 is much larger (approximately 0.1) and noisier. The L2 validation loss for DCSRN however converges at a much smaller value of approximately 0.02.

Given that the DCSRN architecture achieved much better results, we experimented with the effect of using different loss functions when training the network. The performance of the network was evaluated on the test dataset using three different loss functions: L1 loss, L2 loss and perceptual loss. From the metric reported in Table 2, it can be noted that the L1 loss improves performance across all three metrics. When using the L2 reconstruction loss, the network is dis-proportionally driven to correct large differences between the generated and ground truth image. Unlike the L2 loss, the L1 loss will try to match up smaller variations between the two images. The perceptual loss (P) was computed by taking the L1 difference between feature images extracted from a pre-trained VGG network. While the network performance using perceptual loss was better compared to using L2 loss, qualitatively, the network output images lacked some of the finer details compared to the L1 loss output. Figure 7 shows an example output test image, obtained by stitching the patches output by the network back together. Despite the DCSRN model's improved metrics, the resulting image has a dark appearance, and struggles to learn all the details in the input LR image. Furthermore, the CNN output is sensitive to edge artifacts at the boundaries of the different patches. When stitching the patches back together, 4 pixels along the boundary of each patch were cropped out since the receptive field of neural networks is the strongest surrounding the center of images. Although the network output was trimmed along the edges, this artifact suggests that the network performance is poor near the patch boundaries.

7.3. Generative Adversarial Network (GAN) Analysis

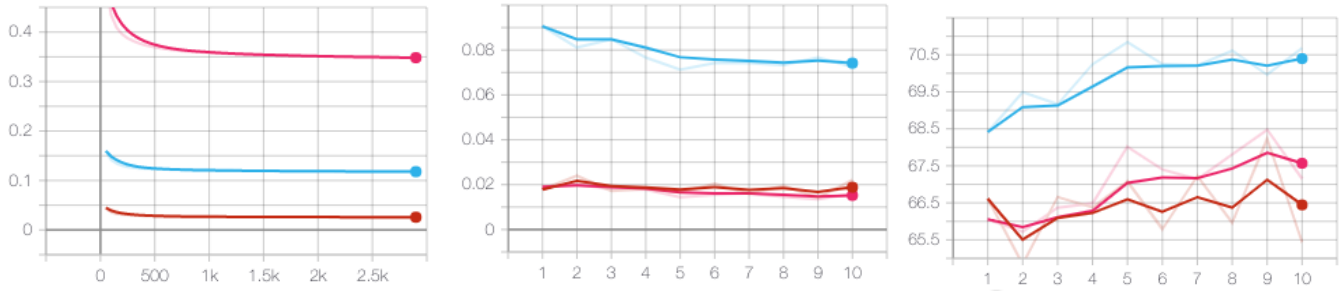
Table 3 presents the evaluation results for the GAN models discussed above. It is worth-noting that the results are relatively consistent for each GAN model. For example, model 2 has the highest PSNR value, the highest SSIM value, the lowest MSE value among all the DCGAN models. Similarly, model 7 and model 9 has the best evaluation results among all the U-net models and all the pix2pix models, respectively. The architecture and the hyper-parameters of each model can be referred to the Table 1.

7.3.1 DCGAN Analysis

As stated above, based on the evaluation table, we found that the images generated by the model 2 are closer to the ground truth (high-resolution images) than those by the other three DCGAN models. Model 2's generator is made up of five layers and has 196 filters at maximum. The standard binary cross entropy loss and the L1 reconstruction loss are the only loss it uses. It's stitched patches can be found in Figure 8. The right-most blue-red colored images indicate the differences between the generated images and the low-resolution images, where blue areas indicate positive values (generated have higher intensity than low-resolution images), and red areas indicate negative values. The deep blue area in (a) and (b) subfigure 8 shows that the generator of the DCGAN model correctly detects the brain structure from the input images and enhance the corresponding pixel intensity while generating the high-resolution images (otherwise, the background should should be deep blue as well). Moreover, the red only appears at the corner or around the edge in the difference im-

CNN Model		PSNR ↑	MSE ↓	SSIM ↑	# Parameters
FCN 1		57.01	0.147	0.116	308545
FCN 2		56.47	0.151	0.0679	155681
DCSRN	**L1**	**65.51**	**0.0236**	**0.358**	**53977**
	L2	62.83	0.0406	0.254	53977
	P	64.1	0.0316	0.211	53977

Table 2: CNN Models Evaluation Results

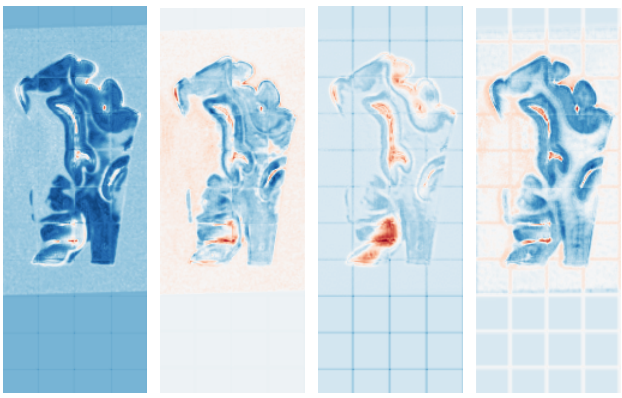


Training Loss | Validation loss | Validation PSNR

Figure 5: Training and Validation curves for DCSRN model, trained using L1 loss (light blue), Perceptual loss (pink) and L2 loss (red).

ages, which means the generator sharpen the images by reducing the intensity at the fuzzy edges. Overall, since the images are stitched up from several small generated images, the lack of apparent stitching edges and inconsistent color means the generator performs consistently and is robust to various input images.

We believe there are two reasons that result in the model 2 surpassing the other DCGAN models: its generator's architecture and its loss function. The model 2's generator has more filters than the model 1's generator so that it could capture more patterns from the low-resolution images, but not as complicated as the model 3 and model 4's generator, which may be over-fitting with too many parameters. The loss function of model 2, and model 3 as well, is a combination of the standard binary cross entropy loss and the L1 reconstruction loss, whereas the loss function of model 1 and model 4 includes an additional gradient loss. The resulting generated images may not be very distinct when we look at single images, but when we stitch small images together to get large patches, patches generated by models with gradient loss has stitching edges on the background that is very hard to ignore. Figure 9 displays the difference images for all four DCGAN models. The two patches on the left are generated by the two models without the gradient loss, while the two patches on the right are generated by the models with gradient loss.

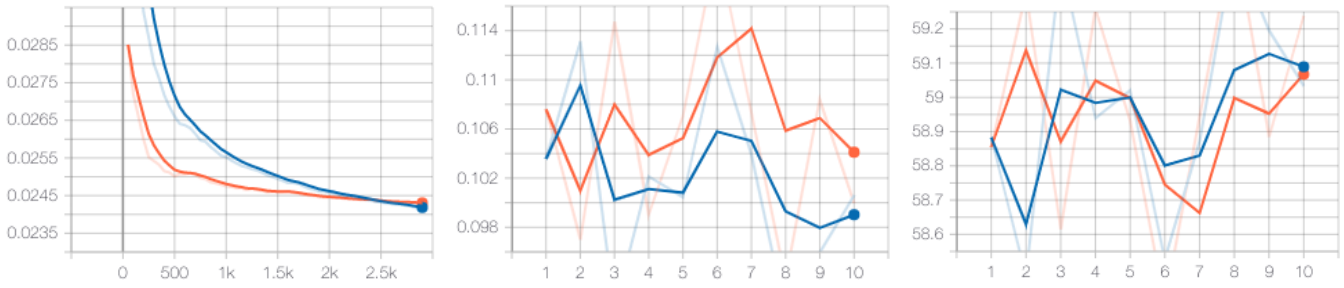


(a) Model 2 (DCGAN) Difference Image | (b) Model 3 (DCGAN) Difference Image | (c) Model 1 (DCGAN) Difference Image | (d) Model 4 (DCGAN) Difference Image

Figure 9: Difference Images for the Four DCGAN Models

7.3.2 U-net Gen Analysis

All the U-net models have the same generator and discriminator architecture, and thus the same parameter numbers. The original generator from the DCGAN models were replaced by the U-net generator with skip connections that concatenate filter maps before the bottleneck with the filter maps after the bottleneck. DCGAN 2 Discriminator served as the discriminator of the new models as well based on our



Training Loss | Validation loss | Validation PSNR

Figure 6: Training and Validation curves for FCN models, trained using L2 loss.

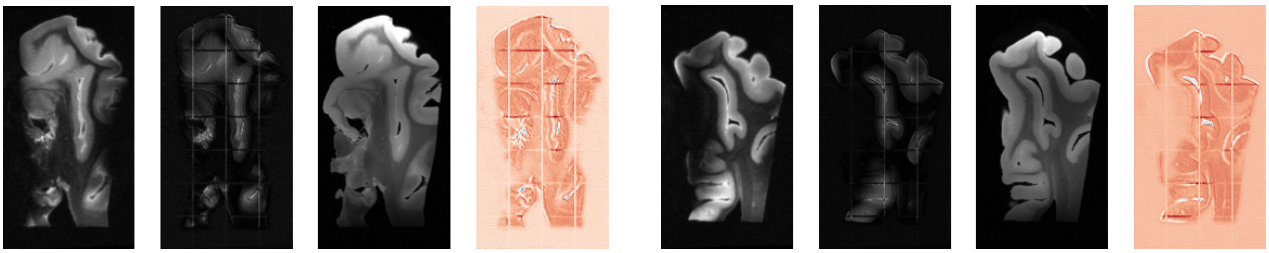


Figure 7: Evaluation Images of the DCSRN L1 Model. From left to right: original low-resolution image, generated high-resolution image, original high-resolution image, and the difference between the generated image and the original low-resolution image, which indicates the degree the model has learned from the training

experiments on the DCGAN. According to the final evaluation results (Table 3), model 7 performs the best among all the U-net models, which means the generated images of model 7 is more similar to the high-resolution images. Despite the same architecture, the main distinction between the model 7 and all the other U-net models is its loss function: it only included the binary cross entropy loss and the L1 reconstruction loss, while the others add an extra gradient loss. Besides, we did not apply a sharpening kernel (like model 6) to model 7. Just as the model 2, the blue ares in the difference image shows in Figure 12 show that the generator distinguished the object from the background and made an effort to accentuate the brain structures while generating the images. The red areas show the U-net generator was trying to getting closer to the high-resolution images by decreasing the vagueness from the low-resolution images. Something notable is that there are many light colored transition area between the blue areas and the red areas. This phenomenon reflects that the generator was not confident about the edges, so it did not change the pixel intensity for those places from the low-resolution images. In addition, there are stitching edges over the object (but not over the background) in the difference images, which may indicate that the U-net generator finds it difficult to deal with the borders of the images.

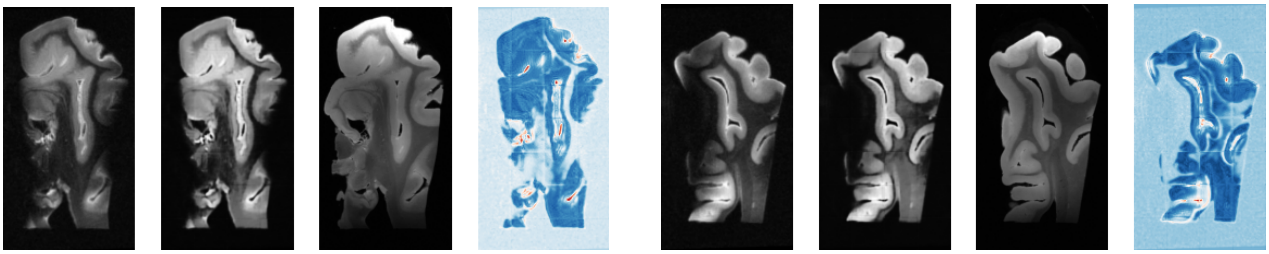
The main reason that model 7 performs better than the other three U-net models is that it did not use the gradient loss. Just as what happened to the DCGAN models, stitching edges also appear on the background of the images of the models that used gradient loss. As shown in the Figure 10, all the three difference images on the right has stitching edges on the background. An notable difference here is between model 5 (b) and model 6 (c). Compared to model 5, model 6 only had one more convolution operation: it convoluted the output images with a sharpening kernel. As the result, the stitching edges on the background caused by the gradient loss reduce significantly. This may indicate that the sharpening kernel forced the gradient loss to pay more attention to the borders and thus alleviate the problem of apparent stitching edges.

7.3.3 Pix2pix Analysis

Overall, the pix2pix models performed more poorly than DCGAN and U-net models. One of the most significant weakness of pix2pix models is the inconsistency of the background. We experimented with the same architecture of a U-net generator and PatchGAN discriminator, part (g)-(h) in Figure 13, but different generator loss function. We

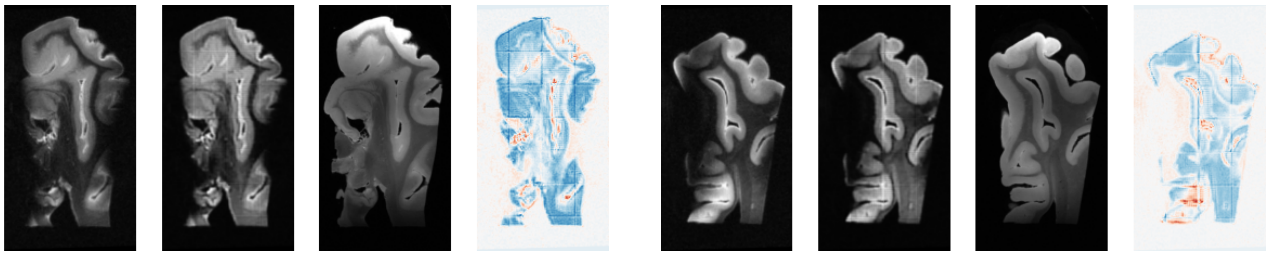
GAN Model		PSNR ↑	MSE ↓	SSIM ↑	# Parameters
DCGAN	1	65.31	0.022	0.255	6885
	2	**67.44**	**0.0178**	**0.578**	**458385**
	3	67.06	0.0194	0.588	868369
	4	65.67	0.0225	0.358	868369
U-net Gen	5	65.90	0.0229	0.500	1829025
	6	66.17	0.0213	0.498	1829025
	7	**67.55**	**0.0174**	**0.593**	**1829025**
	8	66.27	0.0223	0.508	1829025
pix2pix	**9**	**65.87**	**0.0249**	**0.544**	**1829025**
	10	63.18	0.0333	0.237	1829025
	11	61.07	0.0517	0.443	1829025

Table 3: GAN Models Evaluation Results



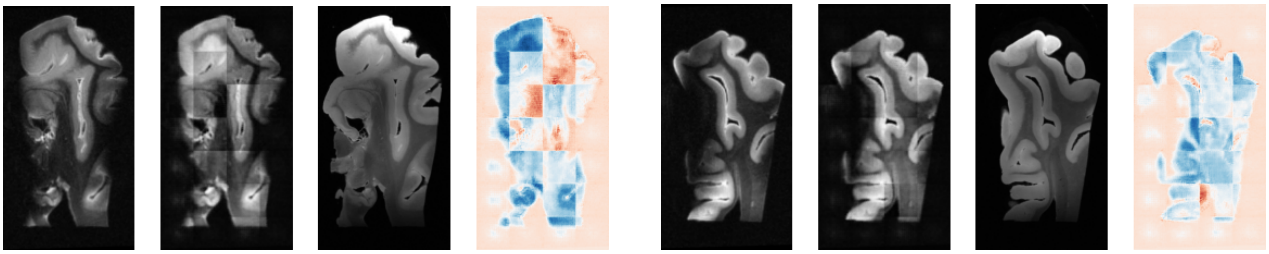
(a) Model 2 (DCGAN) Evaluation Image

(b) Model 2 (DCGAN) Evaluation Image



(c) Model 7 (U-net) Evaluation Image

(d) Model 7 (U-net) Evaluation Image



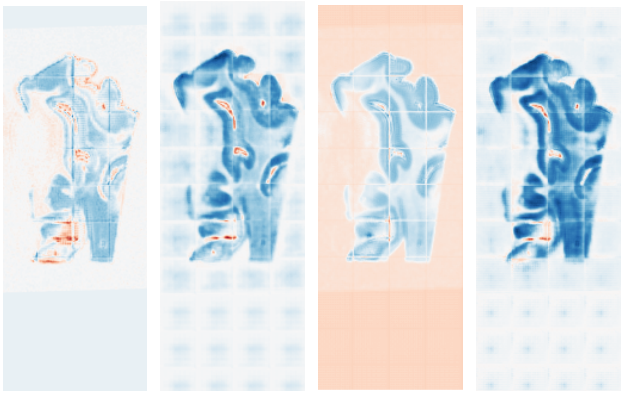
(e) Model 9 (pix2pix) Evaluation Image

(f) Model 9 (pix2pix) Evaluation Image

Figure 8: Evaluation Images of GAN models. From left to right: original low-resolution image, generated high-resolution image, original high-resolution image, and the difference between the generated image and the original low-resolution image, which indicates the degree the generator has learned from the training

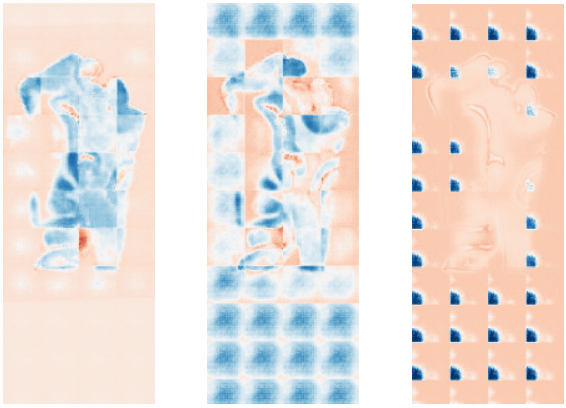
found that using gradient difference loss yielded especially poor results during training, where the generated HR images displayed a strong grid-like structure that doesn't go away with training, So we decided to only experiment with BCE, reconstruction loss and perceptual loss.

The difference between the generated HR and the LR



(a) Model 7 (U-net) Difference Image (b) Model 5 (U-net) Difference Image (c) Model 6 (U-net) Difference Image (d) Model 8 (U-net) Difference Image

Figure 10: Difference Images for the Four U-net Models



(a) Model 9 (U-net) Difference Image (b) Model 10 (U-net) Difference Image (c) Model 11 (U-net) Difference Image

Figure 11: Difference Images for the Three Pix2pix Models Models

images is shown in Figure 11. Model 9 uses the L1 reconstruction loss, model 10 uses the L2 reconstruction loss and model 11 uses the L1 reconstruction loss along with the perceptual loss. At a glance, model 11 performs very poorly when perceptual loss is added, where the generator starts to produces artifacts on the lower left corner for background patches where there's no active brain regions. Those areas with the brain regions in the generated HR images have lower intensity than the original LR images, showing that the perceptual loss fails to generate higher resolution images.

Comparing the L1 and L2 loss, L1 reconstruction loss has shown to perform better in DCGAN and U-net training, and is the same cases here for Pix2pix models. L1 reconstruction loss is able to enhance the active brain region compared to the background, which is our desired result, whereas L2 reconstruction loss enhances the background patches, but reduces the intensity of the active brain regions, see part (a)-(b) in Figure 11. Comparing model 9 and model 10 shows that L1 reconstruction loss allows the generator to generate better images.

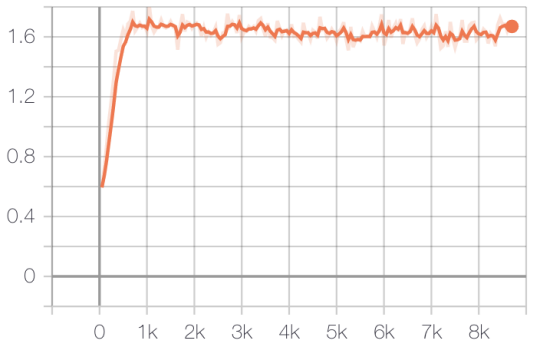
7.3.4 GAN model Analysis

We now want to compare the best models from the DCGAN, U-net, and pix2pix: model 2 (DCGAN), model 7 (U-net), and model 9 (pix2pix). The stitched together patches from these three models can be found in Figure 8. The evaluation result shows that the performance of model 2 and model 7 are about the same, while model 9 performs worse than them. The stitching patches 8 and the loss curve 12 show the same thing. We can recognize integrated brain structures in the stitching patches of model 2 and model 7 because they all have clear edges and consistent color, but we can only make a guess when looking at the stitching patches of model 9 because the small images are so different from each other, which shatters the feeling of a whole. Since all the three models only used the binary cross entropy loss and the L1 reconstruction loss, we can compare their training loss curves directly. Although all the three discriminator losses converge at around 0.1, there is a difference in the generator losses. Model 2's generator loss kept going up and converged around 1.6; model 7's generator loss went up first and decrease shortly, but eventually, it also converged around 1.6; like model 2's generator loss, model 9' generator loss also kept rising, but it converged around 3.2 at the end, which is much larger than 1.6.

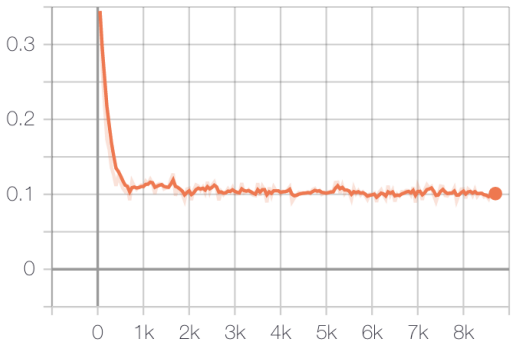
We believe the main reason the pix2pix model performed worse than the other two models is the PatchGAN discriminator. As been described above in the Method section, PatchGAN mainly focus on distinguishing small patches instead of the whole image and its final decision is an average of the small patches. However, small patches can only provide local information. As the result, the discriminator tend to false-fully classify fake images as real or real images as fake. This in turn influence the training of the generator. Figure 12 shows that as the discriminator loss decrease, the generator loss keeps increase. As the discriminator loss reached 0.1, the generator loss also reached 3.2. Finally, in Figure 11, the evaluation difference images, we can see that there are blue area in the background area and red area in the object area, which means the generator was emphasizing the background or ignoring the object.

7.4. Model Comparison

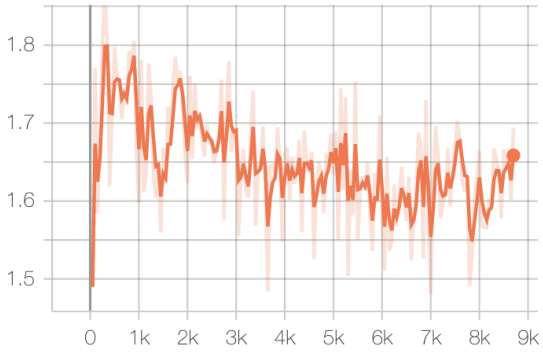
Comparing the results of the three approaches: RFSR, CNNs and GANs, we found that that the GAN models out-



(a) Model 2 Generator Loss Curve

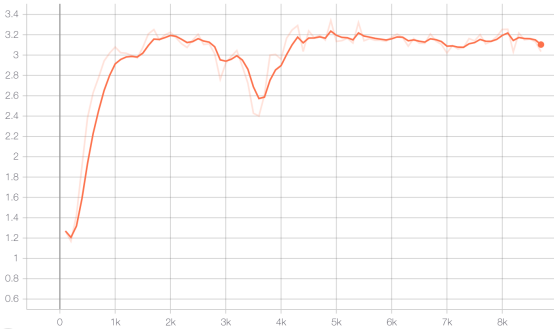


(b) Model 2 Discriminator Loss Curve



(c) Model 7 Generator Loss Curve

(d) Model 7 Discriminator Loss Curve



(e) Model 9 Generator Loss Curve

(f) Model 9 Discriminator Loss Curve

Figure 12: Training Loss Curve for GAN model 1

perform the CNN model, across all three architectures and RFSR. The RFSR output is not distinguishable from the LR input image. Therefore, further quantitative analysis was not done using this apprach. Comparing deep learning models, we observe that using the L1 loss improved our network's performance consistently across all our deep learning experiments. A limitation of our current baseline implementation is that the network architecture used for the GAN generator is not consistent with any of our CNN architectures, thereby making it difficult to directly infer the benefit of incorporating the discriminator network.

With this caveat in mind, from Tables 2 and 3, it can be noted that the DCSRN model, with L1 loss achieved only a slightly lower PSNR compared to the GAN models. Despite this minor difference in PSNR, the GAN models consistently have much higher SSIM metrics. Minimizing the reconstruction loss between two images is equivalent to maximizing PSNR [13]. As mentioned by Ledig *et al.* [11], a higher PSNR does not necessarily provide a perceptually better result. This is clearly observed when looking at our example output test images shown in Figures 7 and 8. Despite the PSNR values for DCSRN with L1 loss and

the best performing GANS being similar, qualitatively, the evaluation images generated by the GAN are significantly better. Unlike the GAN outputs, the CNN output has a much darker appearance, suggesting that the network isn't able to easily distinguish the background and foreground in the image. Given that the best performing networks were all trained using only an L1 loss, this suggests that the improvement obtained by using a GAN is due to the difference in network architecture. We believe that this improvement is likely due to the presence of a discriminator network in the GAN, which drives the generator to output images which are perceptually similar to the ground truth HR images to fool the discriminator.

8. Conclusion & Discussion

For our project, we attempted to generate high-resolution, 9.4T MRI images from lower resolution, 7T MRI images. We evaluated the performance of our GANS against baseline models, namely random forest super-resolution and convolutional neural networks. We also tried several novel approaches to improve the model performance, such as perceptual loss, PatchGAN discriminator, and sharpen kernel. Comparing our results, we found that in general, the GAN models perform better than the former two on this task. In particular, the DCGAN and GAN with U-net generator achieved the best metrics during our quantitative evaluation. Their stitching patches of the generated images shows that the generators of DCGAN and the U-net generator can distinguish the object from the background, render the brain structures consistently, and reduce the vagueness from the low-resolution images.

There are many directions for future work. First of all, the gradient loss did not work well for the borders of the images. This disadvantage got even worse when we tried to stitch together the patches. The borders formed apparent stitching edges which hugely influenced the evaluation of the models. However, during the evaluation of model 6, we notice that introducing the sharpening kernel may have alleviated the problem of stitching edges. We would like to test the relationship between the gradient loss and the sharpening kernel in the future and would also like to experiment with other methods to ameliorate the gradient loss.

In addition to improving the gradient loss, other loss functions used to supervise adversarial learning can also be modified. The instability of GANs during training is well-known, and convergence is not always guaranteed during training. A previous study showed that KL divergence and cross entropy loss are not suitable for training GANS, and proposed using an approximation to the Wasserstein distance [1]. This is another network modification that can be explored in future work.

In this work, we have performed reconstruction on 2D slices, by working slice by slice through a 3D volume, to provide proof-of-concept. This approach does not take advantage of continuous structures in 3D and doesn't have as much spatial context regarding the various anatomical structures. In future work, a 3D GAN can be explored. A 3D network will likely perform better since it can directly extract 3D image features, and consider information across multiple slices.

Finally, our model currently can only enhances the features already in the low-resolution images. This makes the generated images highly depend on the input. If the input image itself does not contain certain features, then the generated image cannot learn the artifacts in the high-resolution images itself. With that in mind, the performance of our GAN could also be further improved by exploring different network architectures and adversarial training schemes. Various generator network architectures have been explored for super-resolution tasks, such as residual networks [17, 5], recursive structures [10, 12] and densely connected networks [3]. Therefore, searching for better model architectures would be a focus of future work.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 13

[2] Khosro Bahrami, Feng Shi, Islem Rekik, and Dinggang Shen. Convolutional neural network for reconstruction of 7t-like images from 3t mri using appearance and anatomical features. In *Deep Learning and Data Labeling for Medical Applications*, pages 39–47. Springer, 2016. 1, 3

[3] Yuhua Chen, Yibin Xie, Zhengwei Zhou, Feng Shi, Anthony G Christodoulou, and Debiao Li. Brain mri super resolution using 3d deep densely connected neural networks. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 739–742. IEEE, 2018. 1, 3, 13

[4] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 13

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. 2, 4, 5

[7] Amod Jog, Aaron Carass, and Jerry L Prince. Improving magnetic resonance resolution with supervised learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 987–990. IEEE, 2014. 1

[8] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. 5

[9] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional net-

works. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. 1

[10] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016. 13

[11] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. pages 4681–4690, 2017. 12

[12] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2019. 13

[13] Dong Nie, Roger Trullo, Jun Lian, Li Wang, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. Medical image synthesis with deep convolutional adversarial networks. *IEEE Transactions on Biomedical Engineering*, 65(12):2720–2730, 2018. 1, 2, 4, 5, 12

[14] Jacob Shermeyer and Adam Van Etten. The effects of super-resolution on object detection performance in satellite imagery. *CoRR*, abs/1812.04098, 2018. 3

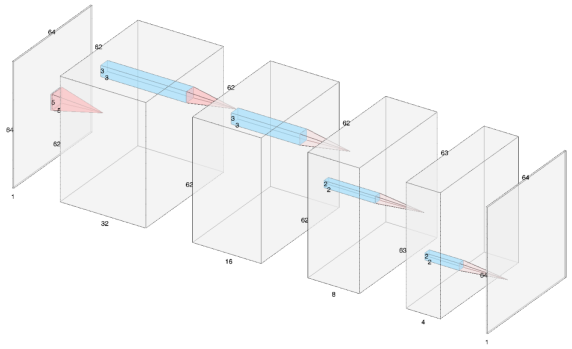
[15] Nicholas J Tustison, Brian B Avants, Philip A Cook, Yuanjie Zheng, Alexander Egan, Paul A Yushkevich, and James C Gee. N4itk: improved n3 bias correction. *IEEE transactions on medical imaging*, 29(6):1310–1320, 2010. 2

[16] Jiancong Wang, Yuhua Chen, Yifan Wu, Jianbo Shi, and James Gee. Enhanced generative adversarial network for 3d brain mri super-resolution. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3627–3636, 2020. 1

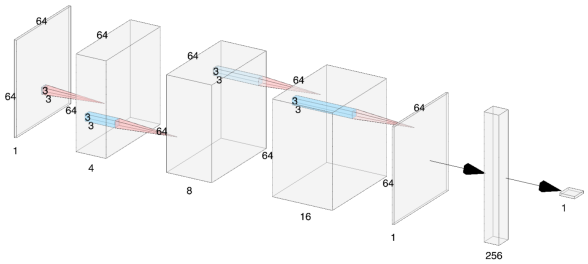
[17] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. 13

[18] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3:47–57, 2017. 5

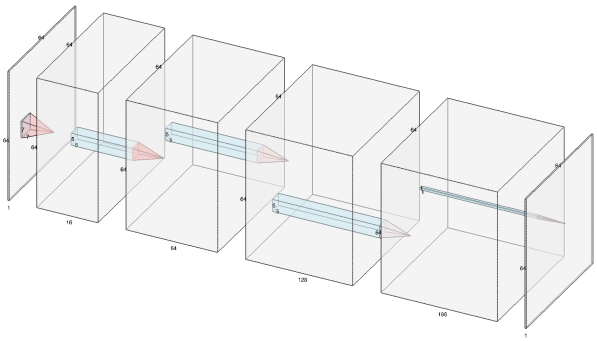
A. GAN Architectures



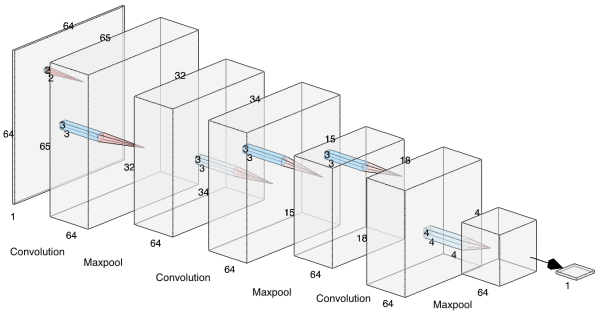
(a) DCGAN 1 Generator



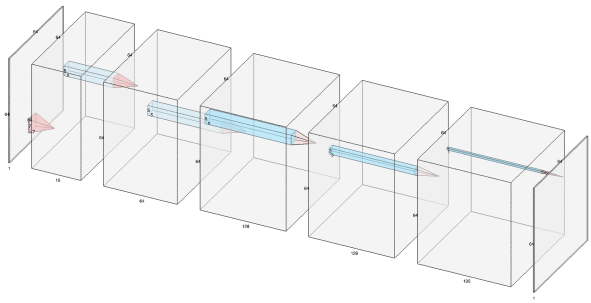
(b) DCGAN 1 Discriminator



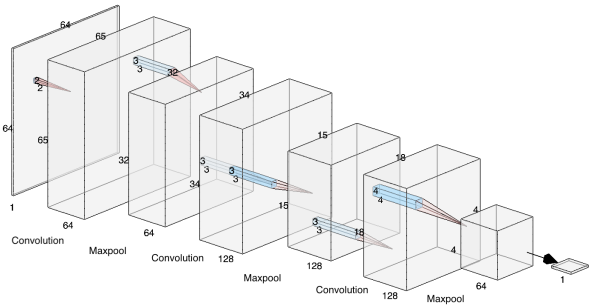
(c) DCGAN 2 Generator



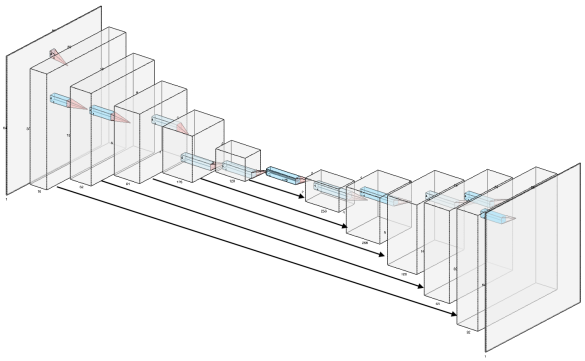
(d) DCGAN 2 Discriminator



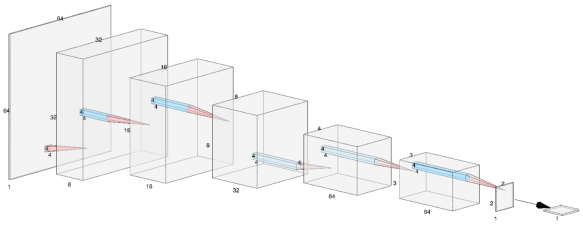
(e) DCGAN 3 Generator



(f) DCGAN 3 Discriminator



(g) U-net Generator



(h) PatchGAN Discriminator

Figure 13: GAN Model Architectures