Semi-supervised Super-resolution GANs for MRI Reconstruction

Lisa Lei
Department of Electrical Engineering
Stanford University
klei@stanford.edu

Morteza Mardani *
Electrical Engineering, Radiology
Stanford University
morteza@stanford.edu

Abstract

Reconstructing high-quality magnetic resonance images (MRIs) is compute-intensive with conventional compressed sensing analytics – image quality has to be sacrificed for real-time imaging. Previous work[1] proposed a supervised deep learning model that was able to construct high-quality MRIs in milliseconds. However, high-quality MRI data that can be used to supervise the training is very limited. In this work we focus on accomplishing the same reconstruction performance with fewer high-quality labels. With our least square generative adversarial network with patching (LSGAN-Patch) model and the Wasserstein GAN with gradient penalty (WGAN-GP)[2] model, we removed from the model in the previous work: the need for pixel-wise supervision, then the need for one-to-one input-label pairing. Without these constraints, we are able to accomplish this specific super-resolution MRI reconstruction task in a semi-supervised manner. Our models are able to reconstruct equally good MRIs when trained with about 1/5 of the labels used in the supervised model.

1 Introduction

MRI has been widely used in clinics and hospitals for medical diagnosis and staging of disease without exposing the subject to radiation. Real-time MRI is essential for clinical practice such as MR-guided neurosurgery [3]. MRI reconstruction is often an ill-posed linear inverse task (i.e. inverse Fourier transform) due to the undersampling in real-time measurements. The obstacles we face are: collecting measurements is time and energy consuming, and rendering clinically-acceptable images from undersampled measurements is computation demanding. We want an algorithm that can reconstruct clinically-acceptable images in a very short time from as fewer measurements as possible.

Previous work [1] demonstrated a fully supervised GANs model that is able to achieve the aforementioned task. Then another challenge follows: neural network training usually required a large amount of data but high-quality MRIs available for research use is very limited. The focus of this work is to extend the existing GANs approach by making it semi-supervised thus reducing the amount of labels needed for training.

At test time, the input to our algorithm is undersampled MRI measurements in frequency domain. We then use a two-dimensional discrete Fourier transform (2D DFT) followed by a generator (i.e. deep residual network[4]) to output a high-quality MRI in time domain. Additionally at train time, the labels to our network (i.e. the discriminator) is the high-quality MRIs in time domain. To push the boundary of the amount of measurements we need, we tried different undersampling ratios and ended up with an undersampling ratio of 3. We also explored the fewest amount of labels needed and ended up with about 1/5 of that used in the fully supervised case.

2 Related work

Traditional compressed-sensing method (CS) utilizes the prior image information to render MRIs from undersampled measurements. However, this method requires solving an iterative optimization that is time-consuming and computationally expensive, and thus not suitable for real-time rendering.

*Provided data, started code and advice. Not enrolled in the class.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

The work in [5] and [6] use the encoder/decoder approach to achieve higher quality reconstruction by denoising, with a speed that is suitable for real-time rendering. But their results suffer from blurry and aliasing artifacts, mainly due the use of pixel-wise L1/L2 costs that is oblivious of high-frequency details.

The authors in [1] have introduced a novel CS framework that permeates benefits from GANs to reconstruct clinically-acceptable images under a few milliseconds at test time. Their model leverages a mixture of least-squares GANs (LSGAN) [7] and pixel-wise L1 loss. Due to the pixel-wise loss, their model needs to be trained in a strictly supervised manner.

A more general approach to generating high-resolution images given low-resolution images using GANs is proposed in [8] using SRGAN. Their work applies to all types of images but is also fully supervised. They defined a perceptual loss function which is critical for the performance of their generator network. All components in the perceptual loss requires one-to-one supervision and one of the component is the L1 loss.

There are unsupervised GANs models like cycleGAN[9] for image-to-image translation and the original GANs[10] for tasks like generator hand-written style digits. But these applications barely demand accuracy on details.

3 Dataset and Preprocessing

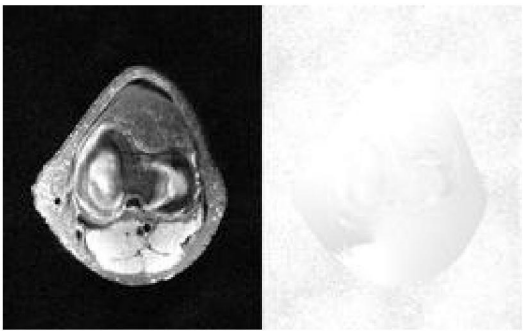

Figure 1: One image for knee MRI, where the left half is the magnitude and right half is the phase.

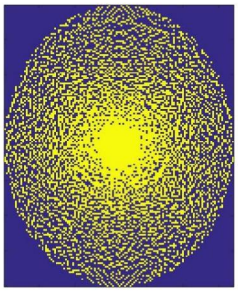

Figure 2: Downsampling mask in frequency domain. Yellow represent ones and blue represent zeros.

The dataset We are using a dataset with high-quality grey scale MRIs for the knee, provided by Stanford Magnetic Resonance Systems Research Laboratory. Knee scans for 17 patients in the training set and scans for 2 patients in the test set. There are 320 jpeg images for each patient. Each image is composed of two parts: magnitude and phase. Figure**??** is one image in our knee dataset, its left half is the magnitude and its right half is the phase. The resolution of each image in the original dataset is 512x320.

Preprocessing We resized the full resolution images in the dataset by Lancoz resampling to 256x160 to reduce the training time. We combine the magnitude and phase extracted from the two halves of each image to a complex valued matrix, normalized the values from 0-255 to 0-1, then split the matrix to two channels holding the real and imaginary part. This is our real input, $\mathbf{x}$, to the discriminator network.

Downsampled inputs: masking in Fourier domain To obtain the undersampled input, we transform the normalized complex valued matrix from the paragraph above to frequency domain, multiple it with the mask, transform it back then split to real and imaginary channels. This is our input, $\tilde{\mathbf{x}}$, to the generator network.

The downsampling mask we finally set according to the capability of our model is shown in Figure 2. This mask gives a downsampling factor of 3 (i.e. 1/3 of this mask are ones).

4 Methods

4.1 Baseline: LSGAN with L1 loss

Our starting point[1] is an extended LSGAN model [1] where the discriminator and generator costs are:

$$L_D(\theta_d) = \mathbb{E}[(1 - D(\mathbf{x}; \theta_d))^2] + \mathbb{E}[(D(G(\tilde{\mathbf{x}}; \theta_g); \theta_d))^2] \quad (1)$$

[1]Starter code: https://github.com/gongenhao/GANCS.git

$$L_G(\theta_g) = (1-\lambda)\mathbb{E}[(1 - D(G(\tilde{\mathbf{x}};\theta_g);\theta_d))^2] + \lambda\mathbb{E}[\ ||\mathbf{x} - G(\tilde{\mathbf{x}};\theta_g)||_1]\ . \tag{2}$$

The last term in the generator cost is the extension part, i.e. the pixel-wise L1 loss between the output of the generator given input $\tilde{\mathbf{x}}$ and the ground-truth label $\mathbf{x}$. The three other terms are standard GANs objectives: we want the discriminator to output ones for ground-truth images zeros for generator outputs, and the generator to get ones from the discriminator. The hyperparameter λ controls how much of L1 loss to use for the generator.

Comparing to the standard GANs [10] that uses cross-entropy losses, LSGAN suffers less from vanishing gradients and is more stable during training. Standard GANs tends to pull samples away from the decision boundary while LS cost instead pulls the generated samples towards the decision boundary [7]. GAN has the problem of introducing high frequency noise all over the image [1]. L1 criterion has proven well in discarding the noise from natural images as it does appropriately penalize the low-intensity high-frequency noise [21].

4.2 LSGAN-Patch

The baseline model diverges without the L1 loss term. Here we describe a modification that stabilizes the LSGAN without pixel-wise loss. Instead of feeding the whole image generated by the generator to the discriminator and get one score as the feedback, we divide the output to 4x4 patches, feed 16 patches separately to the discriminator and average the 16 scores from the discriminator. We do the same for the label. This way the two networks get to focus on smaller images with less complicated details and learn lower level distributions. Equation 3 and 4 defines the objective for this LSGAN with patching model,

$$L_D(\theta_d) = \frac{1}{16}\sum_{i=1}^{16}\mathbb{E}[(1 - D(\mathbf{x}_i;\theta_d))^2] + \frac{1}{16}\sum_{i=1}^{16}\mathbb{E}[(D(G(\tilde{\mathbf{x}}_i;\theta_g);\theta_d))^2] \tag{3}$$

$$L_G(\theta_g) = \frac{1}{16}\sum_{i=1}^{16}\mathbb{E}[(1 - D(G(\tilde{\mathbf{x}}_i;\theta_g);\theta_d))^2] \tag{4}$$

where $\mathbf{x}_i$ is one of the patches. The number of patches to use is a hyperparameter to experiment with.

4.3 WGAN-GP

Apart from the patching approach for the LSGAN, we implemented[1] the Wasserstein GAN with gradient penalty (WGAN-GP)[2] and conclude that the WGAN-GP objective works for our task without modification. The training objectives in WGAN-GP for the discriminator and the generator are as follows:

$$L_D(\theta_d) = \mathbb{E}[D(G(\tilde{\mathbf{x}};\theta_g);\theta_d)] - \mathbb{E}[D(\mathbf{x};\theta_d)] + \eta\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbf{P}_{\hat{x}}}\left[(||\nabla_{\hat{\mathbf{x}}}D(\hat{\mathbf{x}})||_2 - 1)^2\right] \tag{5}$$

$$L_G(\theta_g) = -\mathbb{E}[D(G(\tilde{\mathbf{x}};\theta_g);\theta_d)], \tag{6}$$

where the last term in $L_D(\theta_d)$ is the gradient penalty. Under an optimal discriminator (discussed below), the losses[11] except for the gradient penalty term are derived to minimize the Wasserstein-1 distance between the underlying ground-truth data distribution and the model distribution implicitly defined by the outputs of the generator. The gradient penalty term introduced in [2] improvs the training stability, where η is a constant (by default, $\eta = 10$), and $\tilde{\mathbf{x}}$ represents random samples from a distribution $\mathbf{P}_{\hat{x}}$, which we will describe in the following paragraphs.

Theoretically, WGAN requires an "optimal" discriminator, which has gradients with norm 1 almost everywhere (i.e. being a 1-Lipschtiz function) to effectively minimize the Wasserstein-1 distance. To enforce the Lipschtiz constraint, the original WGAN used gradient clipping. However, the authors in [2] pointed out that this clipping method suffered from problems like vanishing and exploding gradients and proposed to use the gradient penalty term instead as described in Equation (5). To constrain the magnitude of the gradient, they implicitly defined $\mathbf{P}_{\hat{x}}$, sampling uniformly along straight lines between pairs of points sampled from the ground-truth data distribution and the generator output's distribution.

5 Experiments and Results

Our model is implemented in Tensorflow[12]. We use Adam optimizer [13] for experiments unless specified otherwise. We find the default learning rate of 10^{-6} in the starter code is too small and 10^{-4} is a good choice

[1]Reference code: https://github.com/igul222/improved_wgan_training

for the LSGAN models: it speeds up the training a lot without making it unstable. For the LSGAN-Patch model, we use the default hyperparameters for the Adam optimizer. For the WGAN-GP model, we use $= 10$, a learning rate of 5×10^{-5} and $\beta_1 = 0.5, \beta_2 = 0.9$ for the Adam optimizer as sugguested by [2]. Due to the Nvidia K80 GPU memory constraint, the starter code uses a batch size of 2, after resizing the images, we are able to use a mini-batch size of 8.

5.1 Generator and discriminator architecture

The generator is a deep residual network [4] with 4 residual blocks followed by 3 convolution layers and 64 3x3 feature maps for each layer.

We use a data-consistency layer at the end of the generator, this is crucial to the stabilizes our generator when no pixel-wise supervision is present. Equation 7 defines the data-consistency layer output (i.e. the final output of the generator) when the output from the second last layer of the generator is $\mathbf{x}_{-1}$ and $\mathcal{F}$ represents 2D DFT.

$$G(\hat{\mathbf{x}}) = \mathcal{F}^{-1}\{mask \odot \mathcal{F}\{\hat{\mathbf{x}}\} + (1 - mask) \odot \mathcal{F}\{\mathbf{x_{-1}}\}\} \tag{7}$$

The discriminator consists of 7 convolution layers with batch normalization [14] and no pooling. We use 4, 8, 16, 32, 32, 32 and 32 3x3 feature maps for the first to last layer respectively. No fully-connected layer is used. ReLU activation is used after each except for the last layer. Gradient penalty is computed with respect to each input data independently so batch normalization are not compatible with WGAN-GP training. We remove batch normalization layers in the discriminator for the WGAN-GP model.

5.2 Examining the baseline

The first experiment is to see how baseline model ($\lambda = 0.95$ in equation 2) performs. Then for the second experiment, we completely remove the L1 loss by setting $\lambda = 0$. The L1 loss increased rapidly from the beginning and the output images contains nothing besides noise. So for the third experiment, we first train the model with only the L1 loss for 3000 mini-batches to get a good initialization before going to pure GAN training, then set λ to 0 and train the model until the GAN losses converged (we call this model as LSGAN with L1 initialization). The generator L1 loss starts to increase as soon as we set $\lambda = 0$. The generator outputs at test time (Figure 3) is better than that from the first experiment but gets worse as the training goes. Even if the GAN losses converged, the generator output is very bad. We conclude that without the L1 loss, the current model is not able to find a reasonable solution.

We also tried the baseline model without the data-consistency layer in the generator. When the input is the same as the ground truth (i.e. no downsampling) and there is no L1 loss, the generator is not able to generate a reasonable output. So we conclude that besides the L1 loss, the data-consistency layer also plays an important role in the baseline model.

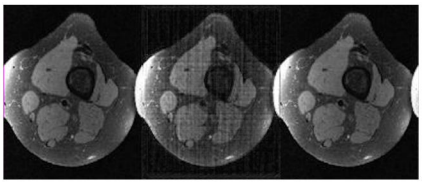

Figure 3: Baseline model when trained with 3000 steps of L1 loss then without L1 loss. Left is input, middle is output, right is ground truth. There are significant artifacts.

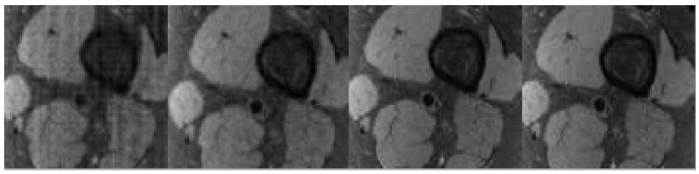

Figure 4: From left to right: output from LSGAN-patch models trained with labels from 3, 6 and 17 patients, ground truth.

5.3 LSGAN-Patch

We started by training the model with 2000 batches with the L1 loss then continue without the L1 loss. Here the L1 loss did not diverge after switching the pure GAN training. Then we reduced the starting L1 training to 200 batches, got the same convergence. Finally we started the training without any L1 batches. The L1 loss increased at the beginning but soon converge to a the order of 0.01, which is same to the convergence value of the baseline model trained with 95% L1 loss.

After removing the L1 loss, we first make sure the model still works well when we break the pairing between the low-quality input image and its high-quality label. Breaking the pairing did not degenerate the performance of our model, as expected, since theoretically, GANs without pixel-wise loss are learning distributions over the whole dataset and order should not matter. Then instead of using high-quality images for all 17 patients as the label to discriminator, we tried training the model using high-quality images from 6 patients and 3 patients,

while the low-quality input to the generator remains the same (i.e. from all 17 patients). Figure 4 shows the zoomed-in output images for the cases where we use labels from 17, 6, and 3 patients. The output from the last case has recognizable artifact, so we conclude that the LSGAN model is able work with about 1/3 of the labels. Quantitative metrics is shown in table 1.

5.4 WGAN-GP

Since we already successfully removed the L1 loss in the LSGAN with patching model, we started experimenting with the WGAN model without any L1 loss, and it worked pretty well. Then we trained the model with different number of labels as done for the LSGAN with patching model. The outputs are shown in Figure 5 and the quantitative scores for these models are shown in Table 1. From the results, it seems that WGAN-GP is more data efficient than the patching model. Qualitatively, the WGAN-GP produces similar images at test time even when training with only about 1/5 of the data.

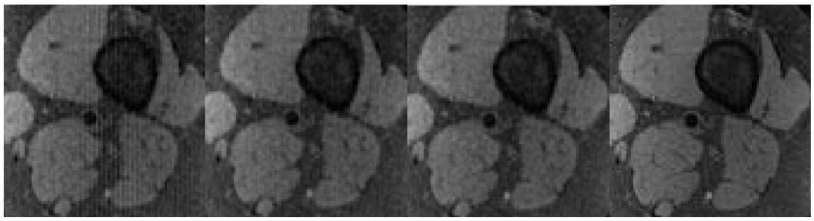

Figure 5: From left to right: output from WGAN-GP models trained with labels from 3, 6 and 17 patients, ground truth.

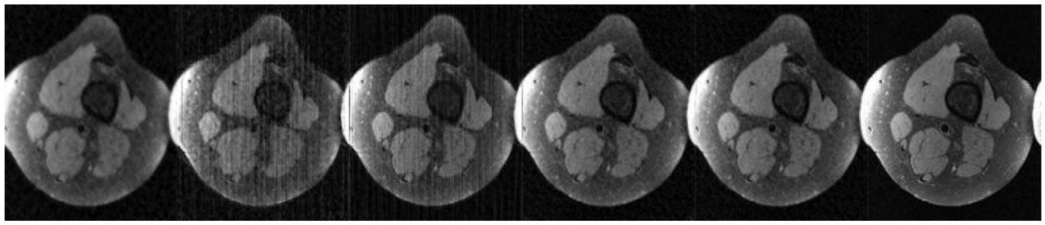

Figure 6: Comparison of various schemes on *full knee dataset* at test time (from left to right): undersampled input (1st); LSGAN with L1 initialization (2nd); LSGAN with L1 pixel-wise penalty (3rd); LSGAN with patches (4th); WGAN-GP (5th); the ground-truth (6th).

Table 1: Average SSIM, SNR comparison of three models with different amount of training labels.

Method	Baseline	LSGAN-Patch	WGAN(3p)[1]	LSGAN-Patch(6p)	WGAN(6p)	WGAN-GP
SSIM	0.76	0.85	0.82	0.81	0.85	0.86
SNR(dB)	18.45	21.41	19.93	20.01	21.01	21.67

[1] with labels from 3 patients. Using labels from all 17 patients if not specified.

6 Conclusion and Future Work

In this work, we demonstrate three GANs models for fast and high-quality MRI reconstruction. We make this specific super-resolution task possible without pixel-wise supervision by using LSGAN-Patch and the WGAN-GP model. When using all labels, even without the pair-wise supervision, both of our models perform better than the baseline with pixel-wise supervision. Then we show that WGAN is slightly better for our application and it can achieve good performance when trained with less than 1/5 of labels used in the supervised approach.

For future work, we would like to try combining patching with the better performing WGAN-GP model, and explore the number of patches as a hyperparameter. We would also try adding feature matching[15] to our generator. Beyond these, we would like to use labels from one part of the body to supervise undersampled input from another part of the body.

References

[1] M. Mardani, E. Gong, J. Y. Cheng, S. Vasanawala, G. Zaharchuk, M. T. Alley, N. Thakur, S. Han, W. J. Dally, J. M. Pauly, and L. Xing, "Deep generative adversarial networks for compressed sensing automates MRI," *CoRR*, vol. abs/1706.00051, 2017.

[2] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

[3] "[online] clearpoint system overview." `http://www.mriinterventions.com/clearpoint/clearpoint-overview.html`.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[5] A. Majumdar, "Real-time dynamic MRI reconstruction using stacked denoising autoencoder," *CoRR*, vol. abs/1503.06383, 2015.

[6] H. Chen, Y. Zhang, W. Zhang, P. Liao, K. Li, J. Zhou, and G. Wang, "Low-dose ct via convolutional neural network," vol. 8, pp. 679–694, 01 2017.

[7] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang, "Multi-class generative adversarial networks with the L2 loss function," *CoRR*, vol. abs/1611.04076, 2016.

[8] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016.

[9] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[15] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016.