CS559 - Deep Learning: Project Report

Beril Alyüz, Aslı Alpman

***Abstract*—CS-MRI aims to reduce the scan time of MRI by using alternative sampling approaches which exploit the redundancies in k-space data. Then, the ill-conditioned inverse problem is solved by enforcing sparsity on some hand-crafted transform domain with iterative complex optimization algorithms to reconstruct the image. It is recently shown that deep learning methods can learn more versatile transformations, which significantly improves the reconstruction results. This project aims to extend ISTA-Net+ architecture, which uses deep learning to learn a sparsifying transformation, to learn two different sparsifying transformations. In this way, we obtained two distinct prior and more information which improved the reconstruction quality.**

***Index Terms*—CS-MRI, Inverse Problem, Deep Learning**

I. Introduction

COMPRESSED Sensing uses the fact that most of the natural signals have a sparse representation in some transformation domain to represent the signals with less number of samples than conventional Nyquist Shannon Sampling [1]. The approach aims to reconstruct a close estimation of the original signals from a small amount of linear measurements by using these sparsifying transforms.

MR images cannot be represented sparsely in the image domain. However, the redundancies in the representations can be eliminated with some techniques and sparse representation can be obtained [2]. For example, total variation (TV) and wavelet transform are used for spatial domain redundancies and parallel imaging methods such as GRAPPA and SENSE are used to remove the coil domain redundancies.

II. Theory and Background

A. Problem Formulation

In MRI, measurement data is the Fourier coefficient of the underlying image. In other words, data is acquired in the Fourier domain, referred to as k-space. The imaging process can be formulated as follows.

$$x = \mathbf{A}y + n \tag{1}$$

This formulation assumes a linear imaging model where x is the ideal image, y is the measurement data, n is the measurement noise, and A is the encoding matrix that establishes the transform between the measurement and image spaces. For MRI, encoding operator A is the multiplication of undersampled discrete Fourier transform operator F_u with the coil sensitivities. Fourier transformation comes from the fact that the measured data corresponds to the Fourier transform samples of the underlying image.

$$\mathbf{A} = \mathbf{F}_u\mathbf{S} \tag{2}$$

B. Alyuz and A. Alpman are with the Department of Electrical and Electronics Engineering, Ihsan Dogramaci Bilkent University, Ankara, 06800 Turkey e-mail: (beril@ee.bilkent.edu.tr), (alpman@ee.bilkent.edu.tr).

One has to solve the inverse problem in equation 1 to reconstruct the image from measurement k-space data. However, the inverse problem is generally ill-conditioned, which means that the solution does not exist, or solution is not unique, or the solution is very prone to the initial conditions. For MRI, we generally undersample to reduce scan time, which resulted in a higher number of unknowns than the number of equations. Thus, it may not be possible to obtain a unique solution.

To uniquely solve those type of problems, we can use regularization using prior information about the underlying image. Then, the resulting minimization problem will enforce the regularization term (regularization step) besides conforming to the measurement (data consistency step) shown as follows.

$$\hat{x} = \min_x \frac{1}{2}||\mathbf{A}x - y||_2^2 + \mathcal{R}(x) \tag{3}$$

In compressed sensing, we assume that the underlying image is sparse in some transform domain. Sparsity means that the transformed image will only have a few nonzero elements. Convex approximate of sparsity constraint is ℓ_1 norm, and thus we want the image has small ℓ_1 norm in that transform domain. As a result, the regularization term and optimization problem take the following form where Ψ is the transformation operator.

$$\hat{x} = \min_x \frac{1}{2}||\mathbf{A}x - y||_2^2 + \lambda||\Psi(x)||_1 \tag{4}$$

Some of the popular sparsifying transforms for the CS-MRI reconstruction problem are discrete wavelet transforms, discrete cosine transform, gradient-domain transforms, etc. Using those hand-crafted transforms makes the theoretical analysis of the optimization problem easier since those transforms have compact inverse forms. The optimization problem is generally solved iteratively with convex optimization algorithms such ADMM, ISTA etc. [3].

Those hand-crafted sparsifying transforms should be engineered carefully since the analytical tractability of the iterative algorithms requires transforms to be linear and invertible. It is also possible to use deep learning methods to obtain the sparsifying transformation or regularizer functions during the training process instead of predefining the transforms and regularizer functions [3]. Our project uses ISTA-Net+ network which learns the sparsifying transformation as backbone architecture. The main contributions of this project can be summarized as follows.

- It learns two information-wise different priors to be utilized in inverse problem regularization.
- It proposes two different loss design approaches (asymmetry loss and inner loss) to learn information-wise different priors in an end-to-end fashion.

III. ISTA-Net+

ISTA-Net+ unrolls the Iterative Shrinkage-Thresholding Algorithm (ISTA) to solve the CS-MRI problem [4]. While it applies the data consistency constraint using the classical ISTA formulation, the regularization block elements (such as nonlinear transforms, shrinkage threshold, and step size) are learned in an end-to-end fashion.

For one iteration of ISTA, the network has one data consistency and consecutive regularization blocks. Data consistency layers enforce the consistency between the reconstructed image and the measurement data. Regularization block enforces the agreement of the reconstructed image with our prior information, which is the sparsity of the image in some domain. Meantime, this block learns the regularization transformation, Ψ in equation 4, which constitutes the prior information.

The network is composed of consecutive convolutional and ReLu layers separated by the soft-thresholding operator. Before soft-thresholding, the network learns and applies a forward transform Ψ. Since we know that the image should have a sparse representation at that domain, we apply soft-thresholding, the proximal operator of ℓ_1 norm, in that domain. After sparsifying with soft-thresholding, we return to the original image domain with another network that applies and learns inverse transform Ψ^{-1}. Figure 1 shows the ISTA-NET+ for N iterations. Note that the network learns and applies different transforms at each iteration which is different than the classical ISTA solution.

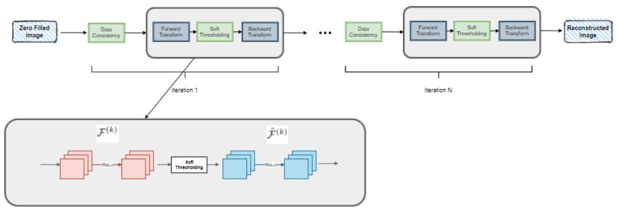

Fig. 1. ISTA-Net+ architecture.

Original ISTA-Net formulates the loss function as follows.

$$\mathcal{L}_{total} = \mathcal{L}_{discrepancy} + \gamma \mathcal{L}_{symmetry} \tag{5}$$

$$\begin{cases} \mathcal{L}_{discrepancy} = \frac{1}{N_b N} \sum_{n=1}^{N_b} ||x_i^{(N_b)} - x_i||_2^2 \\ \mathcal{L}_{symmetry} = \frac{1}{N_b N} \sum_{i=1}^{N_b} \sum_{n=1}^{N_p} ||\tilde{\mathcal{F}}^{(k)}(\mathcal{F}^{(k)}(x_i)) - x_i||_2^2 \end{cases}$$

where N_p is the number of phases (unrolled iterations), N_b is the batch size, N is the number of batches, $\mathcal{F}$ is the forward sparsifying operator and $\tilde{\mathcal{F}}$ is the backward transformation to the image domain.

In this formulation, discrepancy loss ensures that the reconstructed image is consistent with the measurement data, symmetry loss enforce that the network successfully learns the sparsifying transform as well as the inverse of it.

IV. Proposed Model

A. Architecture

We propose to add another branch of transformation to each iteration block such that there is two sparsifying transforms applied to the iteration input. We aim to obtain different inherent information about the iteration input by applying two different transforms where the input has different sparse representation in each transformation domain.

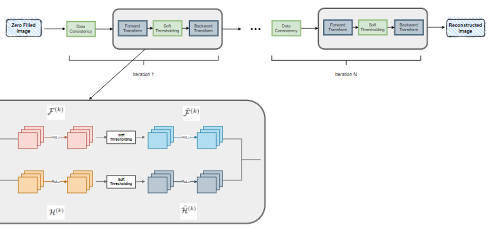

Fig. 2. Proposed architecture.

In order to do this, we kept the data consistency layer as it is but added another branch that go into the data consistency layer from the transformation layer. The structure of the added transformation branch is similar to the original transformation branch in ISTA-Net$^+$ containing a forward transformation block, a soft thresholding operator and a backward transformation operator. Similar to ISTA-Net$^+$, the forward and backward transformation blocks contain 2 convolutional layers with 32 filters with an activation layer between them for nonlinearity. These forward and backward transformation blocks are also learned as in the original ISTA-Net$^+$ architecture. As an attempt to make two transformations distinct from one another, we experimented with the number of filters as well as the activation functions in between them. We tried 16 filters and tanh instead of relu activation. Additionally, in order to enforce distinctiveness between two transformations, additional loss terms are added.

B. Loss Design

1) Asymmetry Loss: The basic idea of the asymmetry loss is to put the iteration input through the first branch's forward transformation and invert it with the second branch's backward transformation. Then compare this result with the original iteration input. If the transformations are distinct, iteration input should be different from the crossed transformation output. Similarly, we can apply the second branch's forward transformation and first branch's backward transformation and compare this output with the original iteration input.

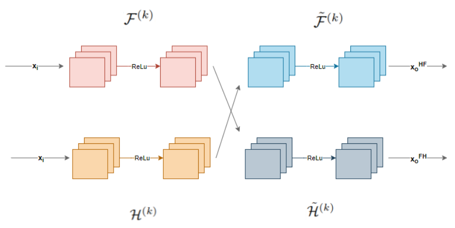

Fig. 3. Asymmetry loss calculation procedure.

There are number of different asymmetry loss formulations that we tried. First we tried mean square formulation of the loss, which is formulated as follows.

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{discrepancy} + \gamma\mathcal{L}_{symmetry} + \beta\mathcal{L}_{symmetry2} \\ - \alpha\mathcal{L}_{asymmetry}\end{aligned} \tag{6}$$

$$\begin{cases}\mathcal{L}_{symmetry2} &= \frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{n=1}^{N_p}||\tilde{\mathcal{H}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2 \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}||\tilde{\mathcal{H}}^{(k)}(\mathcal{F}^{(k)}(x_i)) - x_i||_2^2\end{cases}$$

Then we tried one sided exponential loss, which we take the exponential of the asymmetry loss formulation explained in the first paragraph. It is formulated as follows.

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{discrepancy} + \gamma\mathcal{L}_{symmetry} + \beta\mathcal{L}_{symmetry2} \\ + \alpha\mathcal{L}_{asymmetry}\end{aligned} \tag{7}$$

$$\begin{cases}\mathcal{L}_{symmetry2} &= \frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{n=1}^{N_p}||\tilde{\mathcal{H}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2 \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}e^{-||\tilde{\mathcal{H}}^{(k)}(\mathcal{F}^{(k)}(x_i)) - x_i||_2^2}\end{cases}$$

Thirdly, we tried the two sided exponential loss, where we exponentiate the difference between cross transformed output and the original iteration input with both branch's forward and backward transform. The formulation is as follows.

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{discrepancy} + \gamma\mathcal{L}_{symmetry} + \beta\mathcal{L}_{symmetry2} \\ + \alpha\mathcal{L}_{asymmetry1} + \theta\mathcal{L}_{asymmetry2}\end{aligned} \tag{8}$$

$$\begin{cases}\mathcal{L}_{symmetry2} &= \frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{n=1}^{N_p}||\tilde{\mathcal{H}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2 \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}e^{-||\tilde{\mathcal{H}}^{(k)}(\mathcal{F}^{(k)}(x_i)) - x_i||_2^2} \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}e^{-||\tilde{\mathcal{F}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2}\end{cases}$$

Due to the computational complexity of the exponential function, we wanted to test out the performance of the inverse of the loss where we take the inverse of the difference between original iteration input and the cross transformed output for each forward and backward transform in 2 branches with a slight threshold to avoid division by zero. The formulation of it is as follows.

$$\begin{aligned}\mathcal{L}_{total} = \mathcal{L}_{discrepancy} + \gamma\mathcal{L}_{symmetry} + \beta\mathcal{L}_{symmetry2} \\ + \alpha\mathcal{L}_{asymmetry1} + \theta\mathcal{L}_{asymmetry2}\end{aligned} \tag{9}$$

$$\begin{cases}\mathcal{L}_{symmetry2} &= \frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{n=1}^{N_p}||\tilde{\mathcal{H}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2 \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}\frac{1}{||\tilde{\mathcal{H}}^{(k)}(\mathcal{F}^{(k)}(x_i)) - x_i||_2^2 + \epsilon} \\ \mathcal{L}_{asymmetry} &= \frac{1}{N_b N_p}\sum_{n=i}^{N_b}\sum_{n=1}^{N_p}\frac{1}{||\tilde{\mathcal{F}}^{(k)}(\mathcal{H}^{(k)}(x_i)) - x_i||_2^2 + \epsilon}\end{cases}$$

2) Inner Loss: Inner loss exploits the fact that the inner product is a measure of similarity between two vectors. Since we aim to obtain information-wise different prior, the kernels of the transforms $\mathcal{H}$ and $\mathcal{F}$ should not be similar to each other. Thus, we are taking the inner product of each kernel of one transform with all kernels of the other transform. The matrix $\mathcal{S}$ in figure 4 is composed of inner product values for each kernel combination referred to as the inner product matrix. For example, $\mathcal{S}_{k,t}$ is the inner product of kernel k from $\mathcal{F}$ with kernel t from $\mathcal{H}$. Then, we are summing the absolute value of the resulted inner products as formulated below.

$$\mathcal{L}_{inner} = \frac{1}{N_b N_p}\sum_{n=i}^{N_p}\sum_{i=1}^{N_f}\sum_{j=1}^{N_f}|\mathcal{S}_{i,j}^{(k)}| \tag{10}$$

where N_p is the number of phases (unrolled iterations), N_f is the number of kernels, N_b is the batch size and $\mathcal{S}$ is the inner product matrix.

We are backpropagating through this sum since our purpose is to decrease the similarity between kernels from different transforms and their inner product. One trick that we applied is to normalize each kernel first since the weight updates are greatly affected by scale differences. In this way, we only consider the projection between one kernel to another without depending on their length.

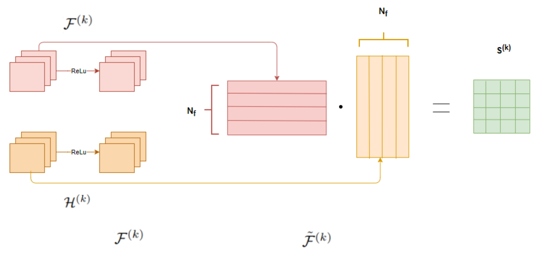

Fig. 4. Calculation of inner loss matrix $\mathcal{S}$.

V. Results

A. Dataset

The original paper uses a custom dataset composed of 800 head MRI images. The dataset is constructed by sampling the given fully-sampled head MRI images with the various compressed sensing ratio. This approach provides the flexibility of training and validating the proposed algorithm with different sampling ratios. We separated 1/8 of the dataset for validation purposes. Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR) is used as validation metrics. While SSIM is bounded with 1 which is an indicator of perfect reconstruction, PSNR is not bounded but higher PSNR indicates better reconstruction.

B. Results

In order to choose the suitable formulation of the asymmetry loss, we compared the performances of the one-sided and two-sided exponential loss by analyzing the constraint loss terms. From Figure 5 and 6, we can see the constraint loss for the one-sided asymmetry loss decreases in more unstabilized fashion than two sided. Therefore, we choose the two sided exponential loss rather than one sided.

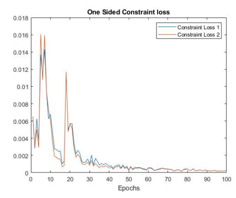

Fig. 5. Constraint Loss for One Sided Exponential Formulation of the Asymmetry Loss

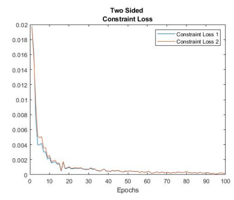

Fig. 6. Constraint Loss for Two Sided Exponential Formulation of the Asymmetry Loss

Then we compared two sided exponential form with inverse exponential form by comparing the constraint losses. The constraint loss of the two sided exponential decreases more rapidly than the constraint loss of the inverse formulation of the asymmetry loss. This might not be advantageous, since with increased epochs, it would become more prone to overfitting. Additionally, the exponential formulation creates a computational complexity which increases the training time. So, the inverse formulation is preferred.

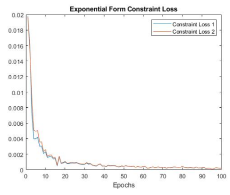

Fig. 7. Constraint Loss for One Sided Exponential Formulation of the Asymmetry Loss

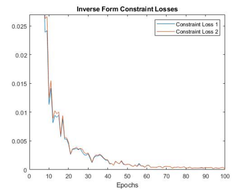

Fig. 8. Constraint Loss for Two Sided Inverse Formulation of the Asymmetry Loss

To learn different transforms, we tried to use different activation functions for different transforms. Figure 9 and 10 shows the results for ReLu-ReLu and ReLu-tanh combinations for the two transform learning network with kernel size as 16. We observed that ReLu is more successful and provides better PSNR and SSIM for all epochs. Since tanh saturated for both larger positive and negative values, it kills more neurons (gradients are zero at saturated regions) and thus, its performance is poor compared to ReLu.

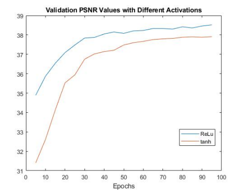

Fig. 9. PSNR comparison of different activation functions.

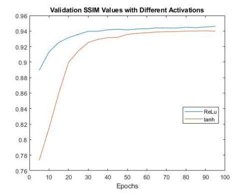

Fig. 10. SSIM comparison of different activation functions.

In order to tune the symmetry loss weights, we tried two different values for symmetry losses 0.01 and 0.05 with kernel size 32. Figure 11 and 12 shows that the symmetry weight 0.01 performs slightly better in later epochs. Although there are not many differences in between, the symmetry weight 0.01 is chosen.

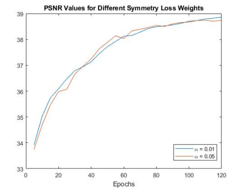

Fig. 11. PSNR comparison of different symmetry loss weights.

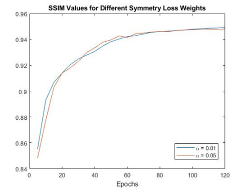

Fig. 12. SSIM comparison of different symmetry loss weights.

To tune asymmetry loss weights (α and μ), we trained the network with $0.001, 0.005$ and 0.0001 with kernel size 16. Figure 13 and 14 shows that the middle value 0.005 gives the best PSNR and SSIM values. We observed that a smaller asymmetry weight causes two transforms to be very similar, while a bigger weight prevents the network from being consistent with the data. Thus, we decided to use 0.005 as the asymmetry loss weight.

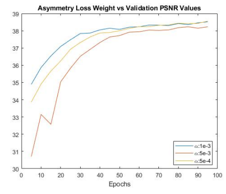

Fig. 13. PSNR comparison of different asymmetry loss weights.

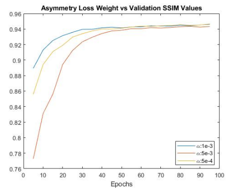

Fig. 14. SSIM comparison of different asymmetry loss weights.

To tune inner loss weights (α and μ), we trained the network with $10^{-6}, 10^{-7}$ and 10^{-8} with kernel size 16. Figure 15 and 16 shows that the middle value 10^{-7} gives the best PSNR and SSIM values. Again, our observation is that a smaller inner weight causes two transforms to be very similar, while a bigger weight prevents the network from being consistent with the data. Thus, we decided to use 10^{-7} as the inner loss weight.

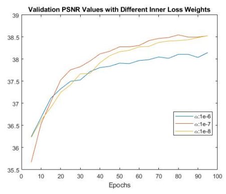

Fig. 15. PSNR comparison of different inner loss weights.

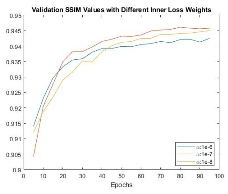

Fig. 16. SSIM comparison of different inner loss weights.

To decide which loss design we are going to use, we compared the performance of each approach. Figure 17 and 18 shows the PSNR and SSIM results for the asymmetry loss or inner loss or combination of them used besides symmetry loss for kernel size 16 For each loss term, tuned weights (as shown previously) are used. In the beginning epochs, we obtained the best results with the inner loss approach. We believe this is because it more directly and explicitly punishes the similarity, while asymmetry loss has a more indirect way of punishing. Therefore, it performs better in the beginning epochs. However, we observed that asymmetry loss and combinational loss performs comparable or even better at the later epochs.

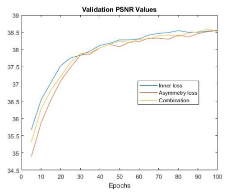

Fig. 17. PSNR comparison of different loss designs.

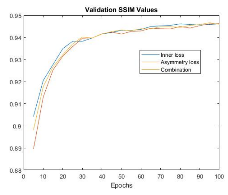

Fig. 18. SSIM comparison of different loss designs.

We also compared the SSIM and PSNR values for different kernel sizes as shown in figure 17 and 18. Since we have limitations in GPU power, most of the tuning is done for kernel size 16. Thus, we compared the performance differences between kernel size 16 and 32, which is the default value in ISTA-Net+. While kernel size 16 performs better in the beginning epoch, kernel size 32 performs better at later epochs. It is reasonable since kernel size 16 is easier to train, and it converges with a lower number of epochs compared to kernel size 32.

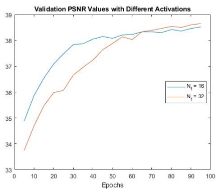

Fig. 19. PSNR comparison of different kernel sizes.

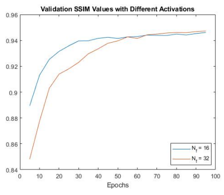

Fig. 20. SSIM comparison of different kernel sizes.

Table I shows the PSNR and SSIM test results for the proposed models together with the original ISTA-Net+ and ADMM-Net. As one can see, our results are comparable with the original ISTA-Net+ and ADMM-Net. One thing to note is that we have comparable result with the original ISTA-Net+ ,which uses 32 kernels, when we are using 16 kernels. In other words, our model achieves comparable results with the original network even if we decrease number of parameters 4 times. We attributed this to the two-fold prior learning.

Another thing to note is that we compared the test results with 200 epochs of training for fair comparison. However, we believe that we would obtain better results for asymmetry loss network with $N_f = 32$ for higher number of epochs. Since our model has higher number of parameters for the same kernel size (due to the second transformation weights), it may require higher number of epochs for training.

TABLE I
TEST RESULTS.

	PSNR	SSIM
Direct reconstruction	30.41	0.7229
Inner Loss ($N_f = 16$)	38.27	0.9451
Asymmetry Loss ($N_f = 16$)	38.30	0.9454
Asymmetry Loss ($N_f = 32$)	38.59	0.9478
ADMM-Net	37.17	-
ISTA-NET+	38.70	0.9484

VI. CONCLUSION

This project aims to learn and apply two information-wise distinct transformations to solve the CS-MRI problem. We obtained comparable results with the backbone architecture ISTA-Net and ADMM-Net. Overall, the idea of learning two different prior or exploiting data to learn information-wise different priors as much as possible may be extended to other deep learning techniques used to solve inverse problems.

REFERENCES

[1] I. Orović, V. Papić, C. Ioana, X. Li, and S. Stanković, “Compressive Sensing in Signal Processing: Algorithms and Transform Domain Formulations,” Mathematical Problems in Engineering, vol. 2016, pp. 1–16, 2016.
[2] J. C. Ye, “Compressed sensing MRI: a review from signal processing perspective,” BMC Biomedical Engineering, vol. 1, no. 1, 2019.
[3] “Deep Magnetic Resonance Image Reconstruction: Inverse Problems Meet Neural Networks,” IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/8962949. [Accessed: 17-Feb-2021].
[4] J. Zhang and B. Ghanem, “ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.